Universidad Autónoma de San Luis Potosí

Facultad de Ingeniería

Área de Ciencias de la Computación

Tecnologías Informática

Ing. Luis Guillermo Canto Sustaita

# **Xtreme programming | Scrum | Kanban**

Alumnos:

Castillo Juárez Alicia Joselyn

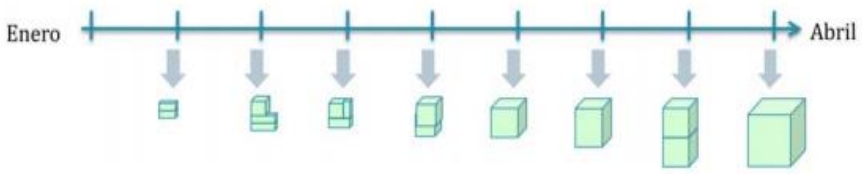
Ortiz Tobías Jessica de Jesús

Marván Media Raúl

# SCRUM

Es una herramienta de proceso que ayuda a trabajar de una forma más eficiente.

- Divide la organización en equipos pequeños.
- Divide el trabajo en una lista de entregables pequeños y concretos.
- Divide el tiempo en iteraciones cortas.



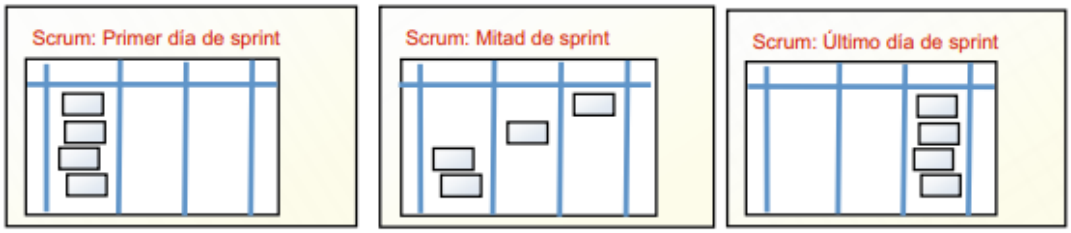

- Optimiza el plan de entregas y actualiza las prioridades.
- Optimiza el proceso.
- Es una herramienta empírica
    - Scrum dice que debemos tener equipos multidisciplinares. Entonces, ¿Quién debe estar en cada equipo? No lo sé, experimenta.
    - Scrum dice que el equipo selecciona cuanto trabajo incluir en un sprint. Entonces, ¿Cuánto deben incluir? No lo sé, experimenta.

- El siclo básico de obtención de feedback es el sprint.

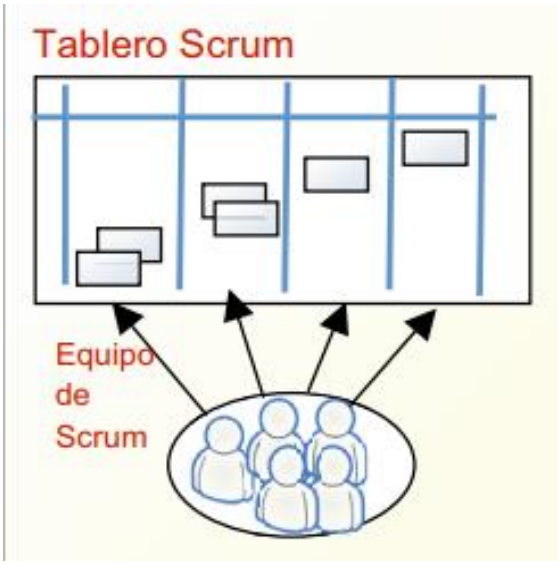

    - El siclo más interno, la programación en parejas, proporciona retroalimentación en unos pocos segundos. Los errores son detectados y corregidos en pocos segundos.
    - El siclo externo, el sprint, proporciona un siclo de retroalimentación de unas cuantas semanas.
    - Un equipo Scrum tiene una reunión corta (de aproximadamente 15 minutos) cada día, a la misma hora y en el mismo lugar. El objeto de estas reuniones es compartir información sobre lo que está pasando,

planificar el día de trabajo actual e identificar cualquier problema significativo.

- El tablero “Sprint” tiene un aspecto similar a este durante sus etapas.




  - Cuando finaliza el sprint, se limpia el tablero.
  - El tablero es solo prioridad de un equipo Scrum.



# eXtreme programming (XP)

El desarrollo de software no es una tarea fácil. Prueba de eso es que hay muchísimas metodologías propuestas que afectan diferentes dimensiones del proceso de desarrollo.

La metodología ágil más popular en la actualidad: eXtreme Programming.

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck es considerado el padre de XP.
Las características esenciales de XP se pueden organizar en los tres apartados siguientes: historias de usuario, roles, proceso y prácticas.

## Las Historias de Usuario

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

## Roles XP

Aunque en otras fuentes de información aparecen algunas variaciones y extensiones de roles XP, en este apartado describiremos los roles de acuerdo con la propuesta original de Beck.

**Programador**

El programador escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

**Cliente**

El cliente escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en

aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.

**Encargado de pruebas (Tester)**

El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

**Encargado de seguimiento (Tracker)**

El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones. También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.

**Entrenador (Coach)**

Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

**Consultor**

Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.

**Gestor (Big boss)**

Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

## PROCESO XP

Un proyecto XP tiene éxito cuando el cliente selecciona el valor de negocio a implementar basado en la habilidad del equipo para medir la funcionalidad que puede entregar a través del tiempo. El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

**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. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

**Fase II: Planificación de la Entrega**

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de

usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

**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. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. 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. Wake en [18] proporciona algunas guías útiles para realizar la planificación de la entrega y de cada iteración.

**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. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

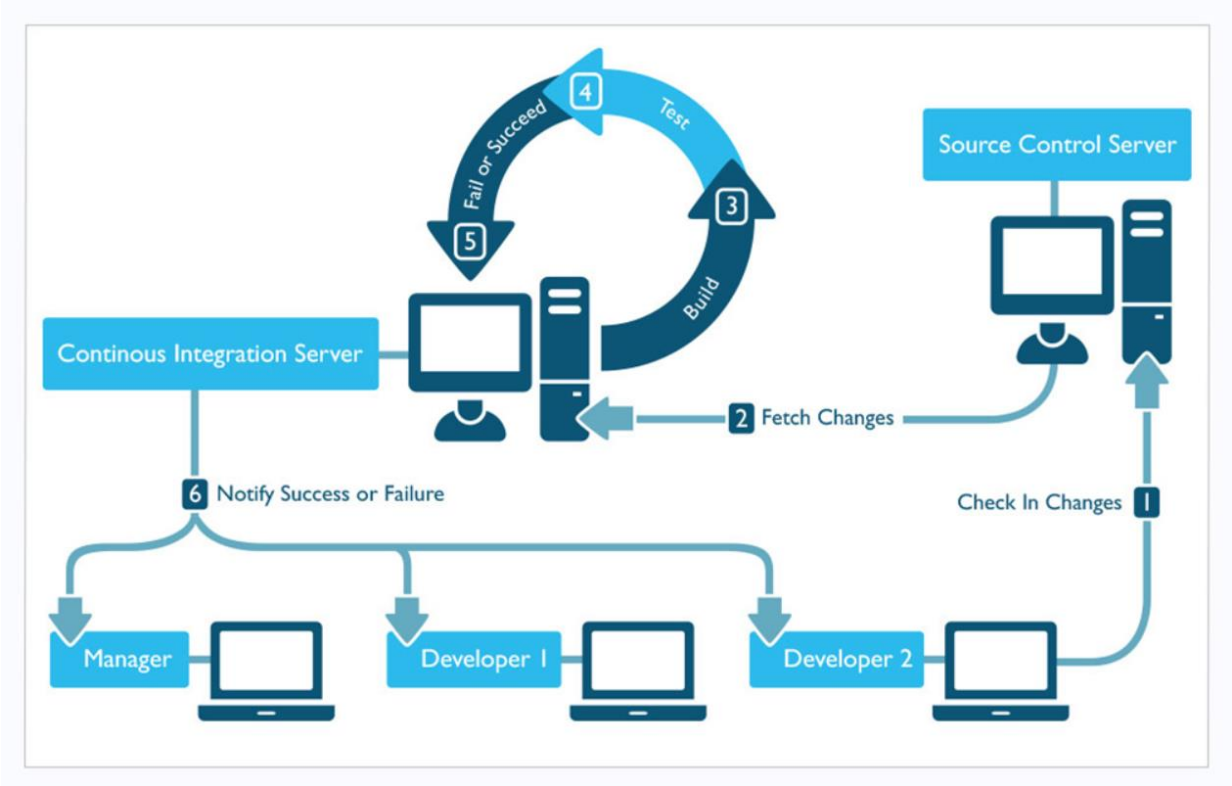
**Fase V: Mantenimiento**

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

### Fase VI: Muerte del Proyecto

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

## Técnica de desarrollo



# KANBAN

En el contexto de la Industria 4.0, el desarrollo de aplicaciones mantenibles y escalables se convierte en una actividad central para dominar para las empresas industriales.

Para ofrecer capacitaciones convincentes, las fábricas de aprendizaje deben coordinar el desarrollo de las soluciones de TI y hardware con los conceptos de capacitación. El desarrollo de soluciones de TI sostenibles, modulares y estables en coherencia con el hardware es la base para una buena capacitación.

El interés en el desarrollo de productos con enfoque Kanban en particular ha aumentado con los años. Sin embargo, los profesionales, en el campo del desarrollo de software, tienen desafíos importantes en la implementación del enfoque Kanban, ya que carece de una definición clara de sus principios, prácticas, técnicas y herramientas. Este estudio tiene como objetivo proporcionar información sobre el enfoque de Kanban y sus elementos (conceptos, principios, prácticas, técnicas y herramientas) que han sido informados empíricamente por académicos y profesionales.

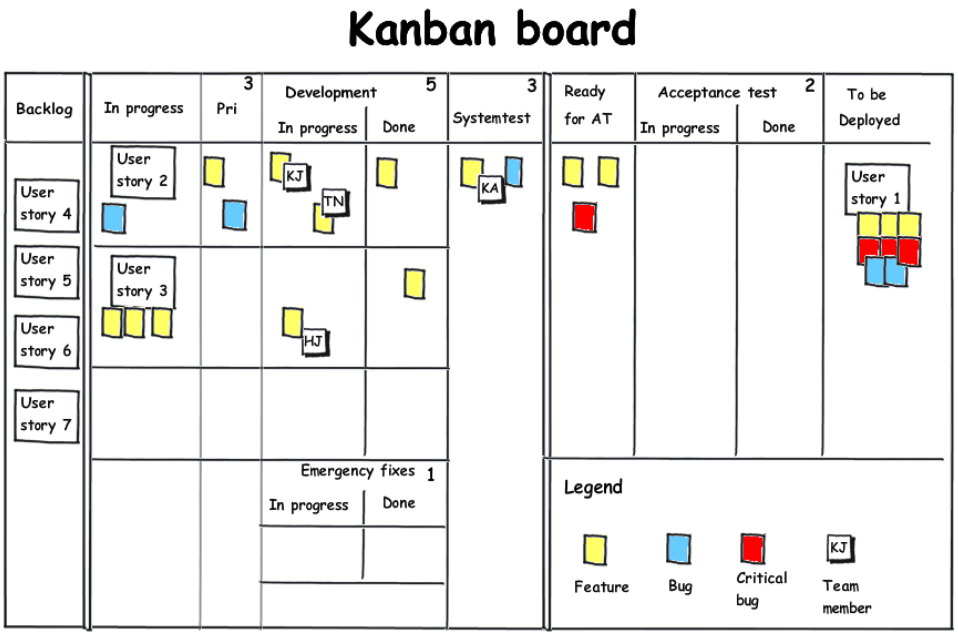
Kanban ha logrado grandes beneficios y mejoras para los equipos de desarrollo de software. Estos beneficios junto con los desafíos se han informado en este estudio. Debido a la variedad de tipos de organización, contextos y tamaños de proyectos informados en los estudios primarios, se espera que los resultados de este estudio ayuden a establecer el conocimiento sobre cuáles son los diferentes elementos del enfoque Kanban, así como a ofrecer un primer paso. Hacia el desarrollo de pautas para que los profesionales ayuden en la introducción del enfoque Kanban a las organizaciones de desarrollo de software.

**Los principios de la metodología Kanban**

La metodología Kanban se basa en una serie de principios que la diferencian del resto de metodologías conocidas como ágiles:

- Calidad garantizada. Todo lo que se hace debe salir bien a la primera, no hay margen de error. De aquí a que en Kanban no se premie la rapidez, sino la calidad final de las tareas realizadas. Esto se basa en el hecho que muchas veces cuesta más arreglarlo después que hacerlo bien a la primera.

- Reducción del desperdicio. Kanban se basa en hacer solamente lo justo y necesario, pero hacerlo bien. Esto supone la reducción de todo aquello que es superficial o secundaria (principio YAGNI).
- Mejora continua. Kanban no es simplemente un método de gestión, sino también un sistema de mejora en el desarrollo de proyectos, según los objetivos a alcanzar.
- Flexibilidad. Lo siguiente a realizar se decide del backlog (o tareas pendientes acumuladas), pudiéndose priorizar aquellas tareas entrantes según las necesidades del momento (capacidad de dar respuesta a tareas imprevistas).
- Visualizar todo lo que está ocurriendo en un momento dado. Cada elemento y su estado de avance se ven en el contexto de todo el trabajo, ya se trate de un proyecto o de las operaciones en curso.
- Limitar la capacidad del trabajo en curso, o Work In Progress (WIP). Hay que poner un máximo a la cantidad de tareas que se pueden gestionar al mismo tiempo y los límites visuales del tablero ayudan a percibir físicamente esa limitación de máximos. Puede que no sea muy intuitivo, pero la limitación del WIP consiste precisamente en visibilizar los cuellos de botella para priorizar el trabajo en esas áreas y poder concentrar recursos en resolverlas.
- Mejorar la continuidad del trabajo. En cuanto se termina un elemento, se inicia otra tarea del backlog. Para ello, es fundamental que el backlog esté correctamente administrado, priorizado y categorizado.



Bibliografía


- Kanban y Scrum – Obteniendo lo mejor de ambos
  Henrik Kniberg & Mattias Skarin
  https://s3.amazonaws.com/academia.edu.documents/38825031/KanbanVsScrum_Castellano_FINAL-printed.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1552426559&Signature=sezSW1FEPXguHrx3qLA8PGPvAJM%3D&response-content-disposition=inline%3B%20filename%3DKanban_Vs_Scrum_Castellano_FINAL-printed.pdf

- https://link.springer.com/article/10.1007/s10664-014-9340-x
- https://www.sciencedirect.com/science/article/pii/S2351978918304633
- http://www.cyta.com.ar/ta0502/b_v5n2a1.htm#(11)