METODOLOGÍAS ÁGILES

XP

Ing. Hernán Naranjo

HISTORIA XP

La programación extrema o eXtreme Programming (XP) es un enfoque de la ingeniería de software formulado por Kent Beck, autor del primer libro sobre la materia, Extreme Programming Explained: Embrace Change (1999).

Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone ***más énfasis en la adaptabilidad que en la previsibilidad***.

Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos

Introducción 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.

Extreme Programming (XP)

La metodología XP define cuatro variables para cualquier proyecto de software:

- costo
- tiempo
- calidad
- alcance

Se especifica que, de estas cuatro variables, sólo tres de ellas podrán ser fijadas arbitrariamente por actores externos al grupo de desarrolladores (clientes y jefes de proyecto).

El valor de la variable restante podrá ser establecido por el equipo de desarrollo, en función de los valores de las otras tres

Extreme Programming (XP)

Este mecanismo indica que, ***por ejemplo***, si el ***cliente*** establece el alcance y la calidad, y el ***jefe de proyecto*** el precio, el ***grupo de desarrollo*** tendrá libertad para determinar el tiempo que durará el proyecto.

El ciclo de vida de un proyecto XP incluye, al igual que las otras metodologías, ***entender lo que el cliente necesita, estimar el esfuerzo, crear la solución y entregar el producto final al cliente***. Sin embargo, ***XP propone un ciclo de vida dinámico***, donde se admite expresamente que, en muchos casos, ***los clientes no son capaces de especificar sus requerimientos*** al comienzo de un proyecto.

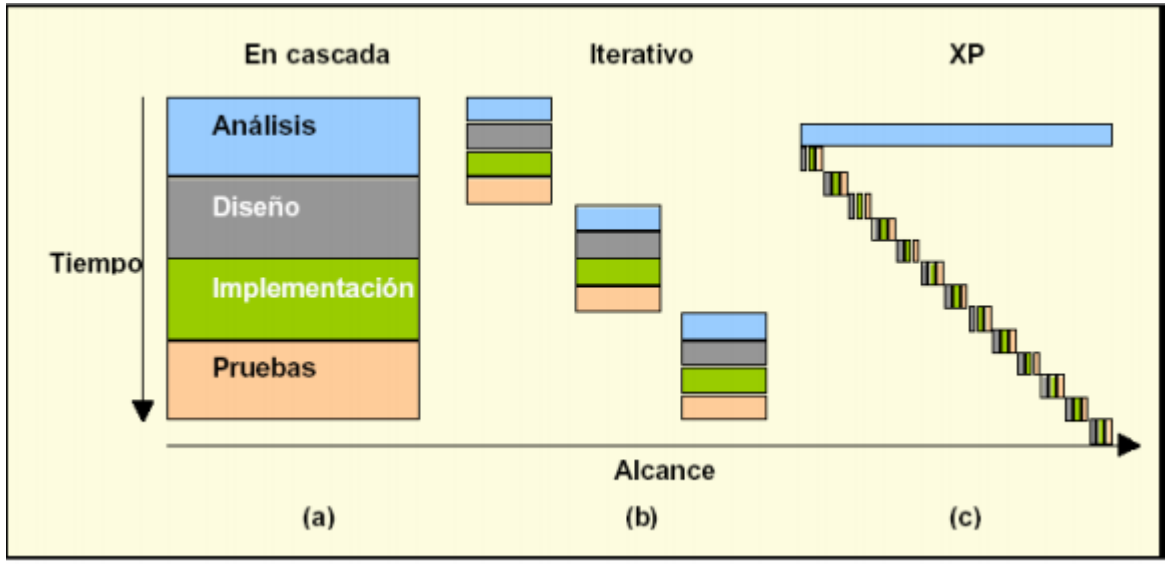
Extreme Programming (XP)

Por esto, se trata de realizar ciclos de desarrollo cortos (***llamados iteraciones***), con entregables funcionales al finalizar cada ciclo.

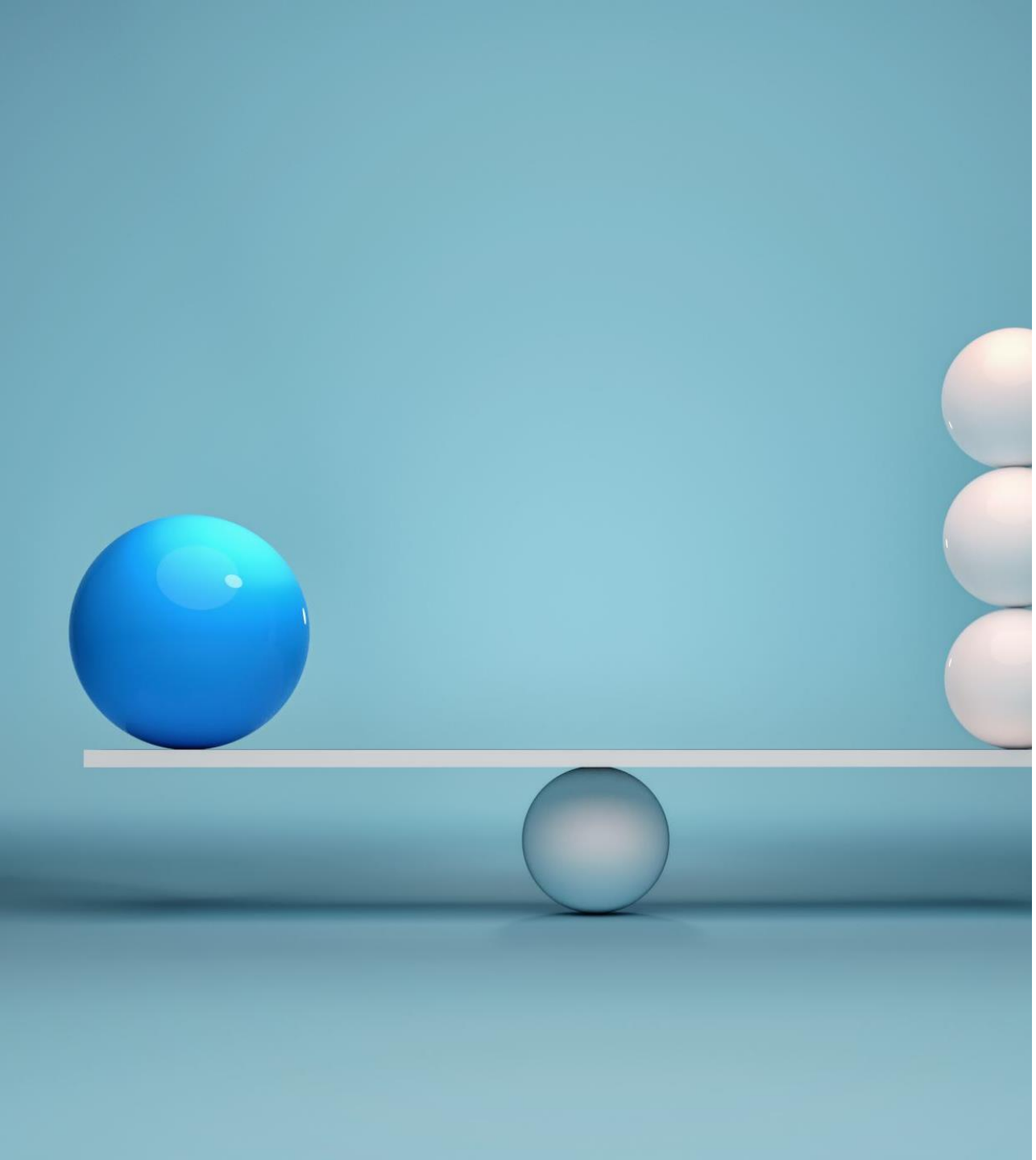
En cada iteración se realiza un ciclo completo de análisis, diseño, desarrollo y pruebas, pero ***utilizando un conjunto de reglas y prácticas*** que caracterizan a XP (y que serán detalladas más adelante).

Típicamente un proyecto con XP lleva ***10 a 15 ciclos o iteraciones***

Extreme Programming (XP)



Si bien el ciclo de vida de un proyecto XP es muy dinámico, se puede separar en fases



Fases

Extreme Programming (XP) / FASES

Fase de exploración

- Es la fase en la que se define el alcance general del proyecto.
- En esta fase, el cliente define lo que necesita mediante la redacción de sencillas ***"historias de usuarios".***
- Los programadores estiman los tiempos de desarrollo en base a esta información. Debe quedar claro que las **estimaciones** realizadas en esta fase son **primarias** (ya que estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más en detalle en cada iteración.
- Esta fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado.

Extreme Programming (XP) / FASES

Fase de planificación

- La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas.
- Típicamente esta fase consiste en una o varias reuniones grupales de planificación.
- El resultado de esta fase es un Plan de Entregas, o "***Release Plan***", como se detallará en la sección "Reglas y Practicas"

Extreme Programming (XP) / FASES

Fase de iteraciones

- Esta es la fase principal en el ciclo de desarrollo de XP.
- Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un ***entregable funcional*** que implementa las historias de usuario asignadas a la iteración.
- Como las historias de usuario ***no tienen suficiente detalle*** como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis***, recabando con el cliente todos los datos que sean necesarios***.
- El cliente, por lo tanto, también debe participar activamente durante esta fase del ciclo.
- Las iteraciones son también utilizadas para medir el progreso del proyecto.
- Una iteración terminada sin errores es una medida clara de avance.

Extreme Programming (XP) / FASES

Fase de puesta en producción

- Si bien al final de cada iteración se entregan módulos funcionales y sin errores, puede ser deseable por parte del cliente ***no poner el sistema en producción*** hasta tanto no se tenga la funcionalidad completa.

- En esta fase no se realizan más desarrollos funcionales, pero pueden ser necesarias tareas de ajuste (“fine tuning”).

Valores

VALORES DE XP

La Programación Extrema “Extreme Programming” (XP) no es un conjunto de reglas a seguir, sino una forma de trabajar en armonía con los valores personales y organizacionales, que tiene su punto de partida en cinco valores fundamentales que son:

- Comunicación
- Simplicidad
- Retroalimentación
- Coraje
- Respeto

Estos valores no son un simple ardid de mercadeo, sino que realmente son parte integral de la metodología.

VALORES DE XP / COMUNICACIÓN

“Todos son parte del equipo y nos comunicamos cara a cara todos los días. Trabajamos juntos en todo, desde los requerimientos hasta la programación. En equipo crearemos la mejor solución al problema.”

En los métodos tradicionales de desarrollo de software, la comunicación de los requerimientos a los desarrolladores se realiza a través de la documentación, por ejemplo las Especificaciones de Diseño en el Rational Unified Process (***RUP***).

VALORES DE XP / COMUNICACIÓN

XP rompe con este esquema, la comunicación se realiza por medio de ***transferencia de conocimientos*** en reuniones frecuentes **cara a cara** entre usuarios y desarrolladores, lo que le da a ambos una visión compartida del sistema.

Por ello***, XP favorece*** diseños simples, colaboración de usuarios con programadores, comunicación verbal frecuente, retroalimentación y construcción rápida del software.

VALORES DE XP / SIMPLICIDAD

“Desarrollaremos lo que se ha solicitado y necesario, pero no más que eso. De esa forma, se ***maximiza*** *el valor de la inversión realizada. Nos dirigiremos a nuestro objetivo a pasos simples y pequeños, mitigando las fallas a medida que ocurran. Crearemos algo de lo cual podamos sentirnos orgullos y que pueda mantenerse en el largo plazo a costos razonables.”*

VALORES DE XP / SIMPLICIDAD

En XP se comienza desarrollando las soluciones más sencillas necesarias para solucionar los problemas (requerimientos) que se están viendo en ese momento, ***añadiendo funcionalidad extra más tarde***, en la medida en que se obtiene más información de los requerimientos.

La diferencia respecto a esquemas tradicionales es que ***se enfoca en las necesidades de hoy*** en lugar de las necesidades de mañana, la semana que viene o el mes que viene.

La ventaja es que no se invierte esfuerzo en futuros requerimientos que podrían cambiar o que no se vayan a necesitar.

Asimismo, un diseño y programación simple mejora la calidad de las comunicaciones, pues es más fácil de implementar y entender por todos en el equipo.

VALORES DE XP / RETROALIMENTACIÓN

"Nos tomaremos seriamente los compromisos con el usuario establecidos en todas las iteraciones, entregando software en funcionamiento en cada una. Mostraremos al usuario nuestro software frecuentemente y de forma temprana, escuchando cuidadosamente sus observaciones y realizando los cambios que sean necesarios. Adaptaremos nuestros procesos al proyecto y no al contrario".

La retroalimentación actúa en tres dimensiones:

Retroalimentación del sistema: Por medio de la ejecución de ***pruebas unitarias y de integración***, los programadores reciben retroalimentación directa del estado del sistema.

VALORES DE XP / RETROALIMENTACIÓN

Retroalimentación del cliente (usuario): Las pruebas de aceptación, son diseñadas conjuntamente por el ***cliente y los analistas de pruebas***, obteniendo en conjunto retroalimentación del estado actual del sistema. Esta revisión puede hacerse cada 2 o 3 semanas, permitiendo así que el cliente sea quien guíe el desarrollo del software.

Retroalimentación del equipo: Cuando el cliente trae nuevos requerimientos, ***el equipo puede directamente proporcionar la estimación del tiempo que tomará implementarlos***.

Bajo este esquema, las fallas de sistema se pueden comunicar fácilmente, pues existen pruebas unitarias que demuestran que el sistema fallará si es puesto en producción. Asimismo, un cliente puede probar el sistema periódicamente, contrastando el funcionamiento con sus requerimientos funcionales o “Historias de usuario”.

VALORES DE XP / CORAJE

"Diremos la verdad en nuestros avances y estimados, no documentaremos excusas para el fracaso, pues planificamos para tener éxito. No tendremos miedo a nada pues sabemos que nadie trabaja solo. Nos adaptaremos a los cambios cuando sea que estos ocurran."

Algunas prácticas del coraje son:

- Diseñar y programar para hoy y no para mañana, evitando así hacer énfasis en el diseño en detrimento de todo lo demás.
- Refactorizar el código siempre que sea necesario (No tener reservas al respecto).

VALORES DE XP / CORAJE

- Inspeccionar constantemente el código y modificarlo (refactorizar), de tal manera que futuros cambios se puedan implementar más fácilmente (desarrollar rápido para atender las necesidades de hoy, pero refactorizar después para facilitar el mantenimiento).
- Desechar componentes o piezas de código cuando sea necesario, sin preocuparse del tiempo invertido (y perdido) en su creación (Es mejor desechar algo que no es útil en lugar de tratar de repararlo).
- Ser persistente en la resolución de problemas.

VALORES DE XP / RESPETO

“Todos en el equipo dan y reciben el respeto que merecen como integrantes del equipo y los aportes de cada integrante son valorados por todos. Todos contribuyen, así sea simplemente con entusiasmo. Los desarrolladores respetan la experticia de los clientes y viceversa. La Gerencia respeta el derecho del equipo de asumir responsabilidad y tener autoridad sobre su trabajo”.

Respeto es tanto por el trabajo de los demás como por el trabajo de uno mismo, por ejemplo: los desarrolladores nunca deben subir cambios que impidan la compilación de la versión, que hagan fallar pruebas unitarias ya realizadas o que de alguna otra forma retrasen el trabajo de sus pares, esto significa tener respeto por el trabajo (y el tiempo) de los demás.

VALORES DE XP / RESPETO

Asimismo, los desarrolladores respetan su propio trabajo por medio de su compromiso con una alta calidad y buscando el mejor diseño para la solución por medio de la refactorización constante.

En cuanto al trabajo en equipo, nadie debe sentirse poco apreciado o ignorado, todos deben colaborar en esto, tratando con respeto a sus compañeros y mostrando respeto por sus opiniones, esto asegura altos niveles de motivación y lealtad hacia el proyecto.

ROLES DE XP

Programador

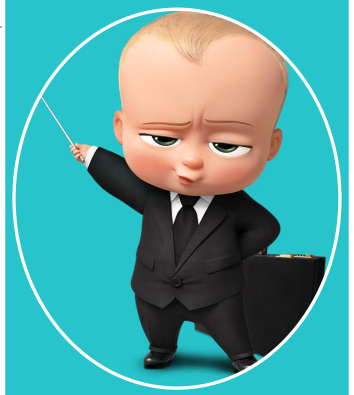
Cliente

Encargado de Pruebas

Encargado de Seguimiento

Entrenador

Consultor



Jefe del Proyecto

ROLES DE XP / PROGRAMADOR

- Pieza básica en desarrollos XP
- Más responsabilidad que en otros modos de desarrollo
- Responsable sobre el código
- Responsable sobre el diseño (refactorización, simplicidad)
- Responsable sobre la integridad del sistema (pruebas)
- Capacidad de comunicación
- Acepta críticas (código colectivo)

ROLES DE XP / CLIENTE

- Pieza básica en desarrollos XP
- Define especificaciones
- Influye sin controlar
- Confía en el grupo de desarrollo
- Define pruebas funcionales

ROLES DE XP / ENCARGADO DE PRUEBA

- Apoya al cliente en la preparación/realización de las pruebas funcionales
- Ejecuta las pruebas funcionales y publica los resultados

ROLES DE XP / ENCARGADO DE SEGUIMIENTO

- Recoge, analiza y publica información sobre la marcha del proyecto sin afectar demasiado el proceso.
- Supervisa el cumplimiento de las estimaciones en cada iteración.
- Informa sobre la marcha de la iteración en curso.
- Controla la marcha de las pruebas funcionales, de los errores reportados, de las responsabilidades aceptadas y de las pruebas añadidas por los errores encontrados.

ROLES DE XP / ENTRENADOR

- Experto en XP
- Responsable del proceso en su conjunto
- Identifica las desviaciones y reclama atención sobre las mismas
- Guía al grupo de forma indirecta (sin dañar su seguridad ni confianza)
- Interviene directamente si es necesario
- Atajar rápidamente el problema

ROLES DE XP / CONSULTOR

- Apoya al equipo XP en cuestiones puntuales

ROLES DE XP / JEFE DEL PROYECTO

- Favorece la relación entre usuarios y desarrolladores
- Confía en el equipo XP
- Cubre las necesidades del equipo XP
- Asegura que alcanza sus objetivos

Herramientas de la metodología XP

Historias de Usuario

Son la técnica utilizada 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.

Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. Las historias de usuario son un artefacto de requerimientos de muy alto nivel.

Historias de Usuario

Beck (Beck, 1999) presenta un ejemplo de Historia de usuario con el siguiente contenido:

Fecha	Tipo de actividad (nueva, corrección, mejora)
Prueba funcional	Número de historia.
Prioridad técnica y del cliente.	Referencia a otra historia previa
Riesgo.	Estimación técnica
Descripción	Notas.
Lista de seguimientos con la fecha.	Estado
Cosas por terminar	Comentarios

Historias de Usuario (ModeloPropuesto)

Historia de Usuario	
Número:	**Usuario:**
Nombre historia:	
Prioridad en negocio: Alta / Media/ baja	**Riesgo en desarrollo:** Alta/Media/Baja
Puntos estimados:	**Iteración asignada:**
Programador responsable:	
Descripción:	
Observaciones:	

Historias de Usuario / Descripción

La descripción escrita es sólo una parte muy pequeña de una historia de usuario. La parte más importante es la conversación, donde el usuario explica al desarrollador qué es exactamente lo que quiere y los detalles de la funcionalidad.

El desarrollador puede escribir anotaciones en la parte trasera del post-it, pero si lo hace tiene que ser una pequeña anotación de pocas palabras.

La forma de redactar la descripción escrita para una historia de usuario es:

Como <rol de usuario>, quiero <función del sistema> para poder <valor de negocio>

Historias de Usuario / Recomendaciones

El cliente escribe las historias del usuario.

Los stakeholders (interesados o involucrados en un problema determinado, en este caso el cliente) del proyecto deben escribir las historias de usuario, no los diseñadores. Las historias de usuario son bastante simples.

Use la herramienta más simple.

Las historias del usuario son escritas en tarjetas de índice. Estas tarjetas son muy fáciles trabajar y son por consiguiente una técnica modelada inclusiva.

Historias de Usuario / Recomendaciones

Indique el esfuerzo estimado

Se debe incluir una estimación del esfuerzo en llevar a cabo la historia de usuario. Una manera de estimar es asignar puntos a cada tarjeta, una indicación relativa de cuánto tiempo tomará que un par de programadores lleven a cabo la historia. El equipo sabe entonces que si actualmente toma 2.5 horas por punto; para una historia que tenga una estimación de 4, tardará alrededor de 10 horas para llevarla a cabo

Historias de Usuario / Recomendaciones

Indique la prioridad.

Los requisitos, incluso defectos identificados como parte de las actividades de comprobación paralelas independientes o por el esfuerzo de operaciones y soporte, son priorizados por los stakeholders de su proyecto

Incluya un único identificador.

Las tarjetas de historias de usuario tienen un único identificador, por ejemplo 173. La razón para hacer esto, es que si necesita podría mantener alguna clase de trazabilidad entre la historia de usuario y otros artefactos, en particular las pruebas de aceptación.

Historias de Usuario / Tareas de Ingeniería

Las **tareas** de ingeniería describen las **actividades** que se realizarán en el proceso descrito en una historia de usuario y se realizan teniendo en cuenta los siguientes datos:

- Está relacionada con el número de historia.
- Se debe poner el nombre de la tarea.
- Los puntos estimados se pondrán dependiendo el tiempo estimado que se llevará en realizar esta tarea.
- Fecha de inicio y fin de la tarea.
- El nombre del programador responsable.
- Descripción de los puntos a tomar en cuenta para realizar la tarea.

Tareas de Ingeniería

Tarea	
Número tarea:	Número historia:
Nombre tarea:	
Tipo de tarea : Desarrollo / Corrección / Mejora / Otra	Puntos estimados:
Fecha inicio:	Fecha fin:
Programador responsable:	
Descripción:	

Pruebas de Aceptación

En una historia de usuario es importante que quede bien definido cómo se va a aceptar, para esto las Pruebas de Aceptación automatizadas son ideales.

Deben ser automatizadas porque deben “abrazar al cambio” (lema fundamental del manifiesto ágil), se debe reaccionar de forma rápida ante él.

Cuando un miembro del equipo está conversando con el usuario sobre una historia de usuario, un tipo de anotación que puede hacer en la cara posterior del post-it es el de recordatorio de cómo testear la historia.

Pruebas de Aceptación

Para obtener una prueba de aceptación se debe seguir los siguientes pasos:

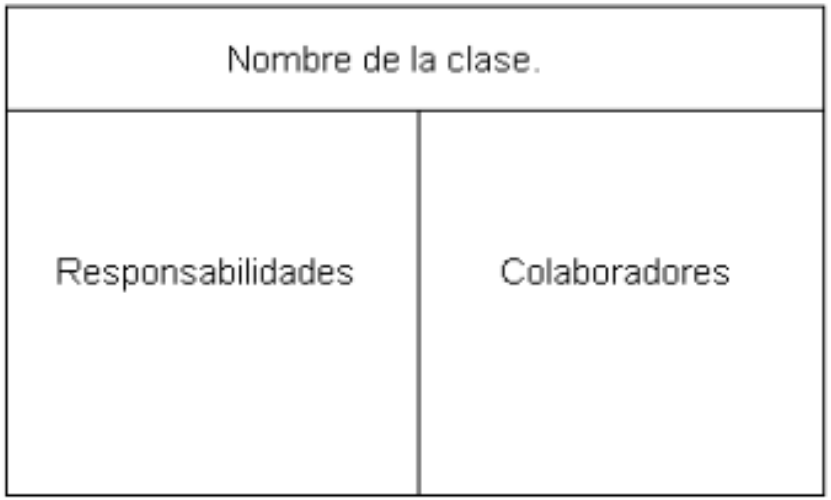
1. Identificar todas las acciones en la historia.
2. Por cada acción escribir dos pruebas.
3. Para algunos datos, suministrar las entradas que debería tener éxito, y llenar cualquier resultado satisfactorio.
4. Para otros datos, suministrar las entradas que tengan una acción fallida, y llenar la respuesta que debería tener.

Pruebas de Aceptación

Caso de Prueba	
Número Caso de Prueba:	Número Historia de Usuario:
Nombre Caso de Prueba:	
Descripción:	
Condiciones de ejecución:	
Entradas:	
Resultado esperado:	
Evaluación:	

Tarjetas CRC

Cada tarjeta CRC representa una clase en la programación orientada a objetos y define sus responsabilidades (lo que ha de hacer) y la colaboración con las otras clases (cómo se comunica con ellas).



Cómo crear modelos CRC

Encontrar las clases.

Es una tarea fundamental porque se identifica los bloques de nuestra aplicación. Una buena regla es buscar de tres a cinco clases principales. Se pueden incluir otras clases inclusive las que representen a los actores.

Encontrar las responsabilidades

Aquí se debe preguntar qué es lo que hace una clase y así saber cuál es la información que se desea mantener de ésta.

Definir los colaboradores

Para identificar los colaboradores para cada responsabilidad debe hacerse la pregunta. ¿Tiene la habilidad para cumplir esta responsabilidad? Si no es así se busca una clase que tenga la habilidad de cumplir la funcionalidad perdida

Prácticas de la metodología XP

Prácticas de la metodología XP

La principal suposición que se realiza en XP es la posibilidad de **disminuir** la mítica curva exponencial del **costo del cambio** a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes prácticas:

Prácticas de la metodología XP

El juego de la planificación. Hay una comunicación frecuente el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.

Entregas pequeñas. Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más de 3 meses.

Prácticas de la metodología XP

Metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una ***historia compartida*** que describe cómo debería funcionar el sistema (conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema).

Diseño simple. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.

Prácticas de la metodología XP

Pruebas. La producción de código está dirigida por las pruebas unitarias. Éstas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema.

Refactorización (Refactoring). Es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo.

Prácticas de la metodología XP

Programación en parejas. Toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño, mayor satisfacción de los programadores).

Propiedad colectiva del código. Cualquier programador puede cambiar cualquier parte del código en cualquier momento.

Prácticas de la metodología XP

Integración continúa. Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.

40 horas por semana. Se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. Si esto sucede, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo.

Prácticas de la metodología XP

Cliente in-situ. El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Éste es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita.

Estándares de programación. XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible

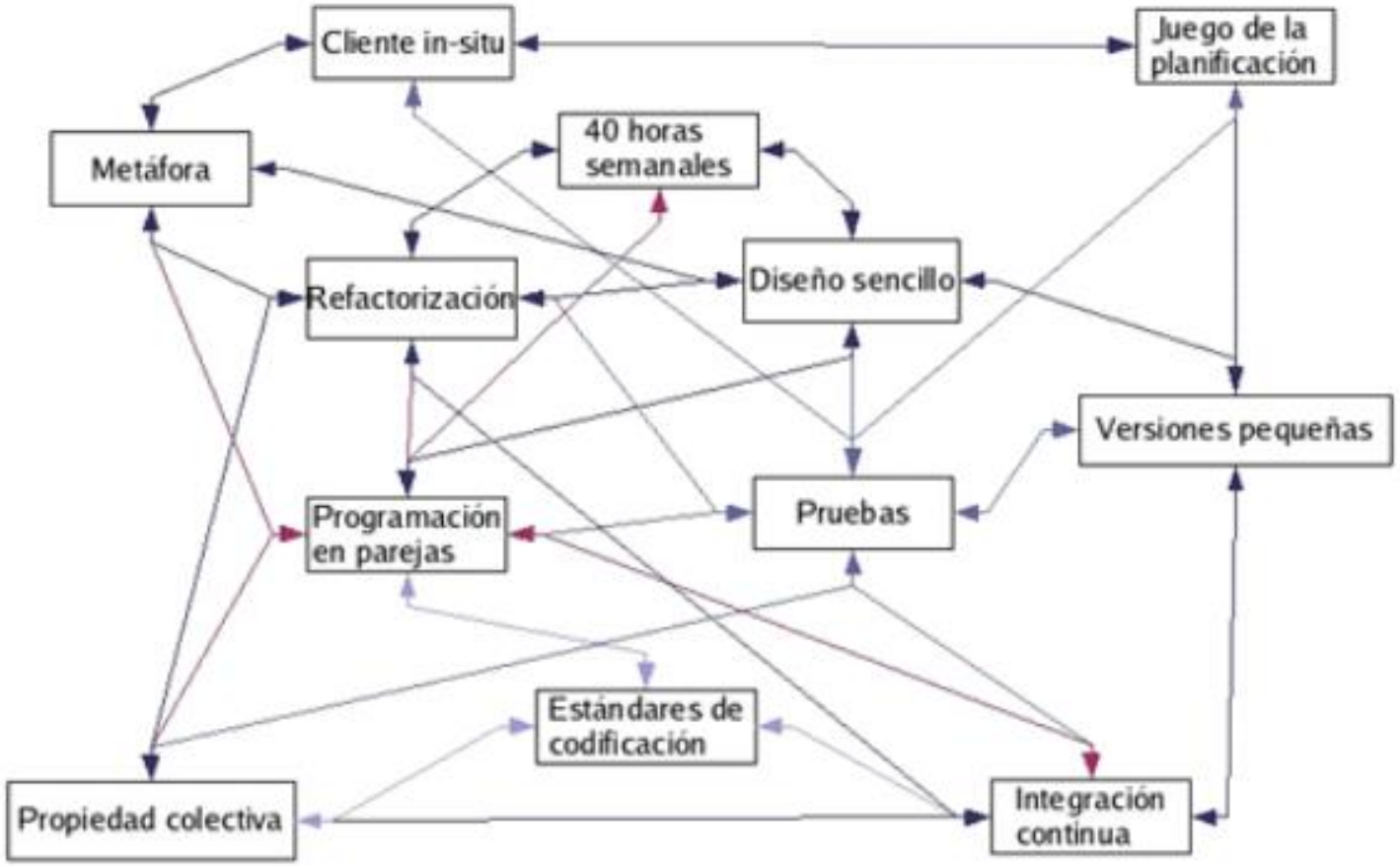
Prácticas de la metodología XP

El mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras.

Esto se ilustra en la figura, donde una línea entre dos prácticas significa que las dos prácticas se refuerzan entre sí.

La mayoría de las prácticas propuestas por XP no son novedosas, sino que en alguna forma ya habían sido propuestas en ingeniería del software e incluso demostrado su valor en la práctica.

El mérito de XP es integrarlas de una forma efectiva y complementarlas con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo.



Prácticas de la metodología XP