Programación Extrema

Índice

1. ¿Qué es XP?

1.1. Metodología ágil

Las metodologías ágiles (como por ejemplo XP, SCRUM, DSDM, Crystal, etc..) forman parte del movimiento de desarrollo ágil de sotfware, que se basan en la adaptabilidad de cualquier cambio como medio para aumentar las posibilidades de éxito de un proyecto.

De forma que una metodología ágil es la que tiene como principios que:

- Los individuos y sus interacciones son más importantes que los procesos y las herramientas.
- El software que funciona es más importante que la documentación exhaustiva.
- La colaboración con el cliente en lugar de la negociación de contratos.
- La respuesta delante del cambio en lugar de seguir un plan cerrado.

Se puede decir que, este movimiento empezó a existir a partir de febrero de 2001, cuando se reunieron los representantes de cada una de estas metodologías y terminaron poniendo en común sus ideas en una declaración conjunta.

1.2. Definición

La programación extrema es una metodología de desarrollo ligera (o ágil) basada en una serie de valores y de prácticas de buenas maneras que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas.

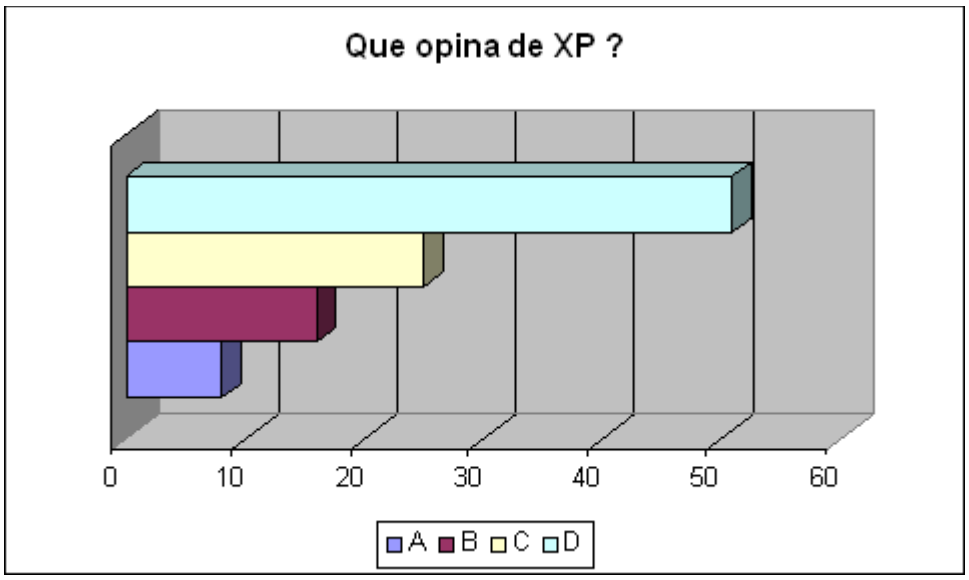
Este modelo de programación se basa en una serie de metodologías de desarrollo de software en la que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay al rededor de la programación.

Una de las características principales de este método de programación, es que sus ingredientes son conocidos desde el principio de la informática. Los autores de XP han seleccionado aquellos que han considerado mejores y han profundizado en sus relaciones y en como se refuerzan los unos con los otros. El resultado de esta selección ha sido esta metodología única y compacta. Por esto, aunque no está basada en principios nuevos, sí que el resultado es una nueva manera de ver el desarrollo de software.

El objetivo que se perseguía en el momento de crear esta metodología era la búsqueda de un método que hiciera que los desarrollos fueran más sencillos. Aplicando el sentido común.

1.3. Posturas a favor y en contra

La mejor manera de reflejar las diferentes posturas sobre las preferencias de la programación Extrema es hacer referencia a una encuesta realizada por IBM el octubre del año 2000, donde se formulaba precisamente la opinión de profesionales sobre el método de programación que nos ocupa.



A . Lo he probado y no me gusta nada

B. Es una mala idea, no puede funcionar nunca

C. Es una buena idea, pero no funcionará.

D. Lo he probado y me gusta mucho

Con el poco tiempo que hace que existe esta metodología, ya se ha generado un gran alboroto en la comunidad de ingeniería del software. Hay opiniones de todos los gustos de quienes lo prueban.

Las opiniones negativas mayoritarias alegan lo siguiente:

- Los programadores tienen un acusado sentimiento de posesión del código y esta postura no encaja con la filosofía de X.P.
- También se ve un fuerte sentimiento para respectar las 40 horas semanales, y X.P. no lo garantiza.
- Los jefes de proyecto también expresan su recelo con este método tan poco tradicional.

En cambio una visión más optimista dice que:

- X.P. sólo funcionará con gente buena, es decir, profesionales que son capaces de hacer un buen diseño, sencillo y a la vez fácilmente ampliable.
- Por otro lado se ha de recalcar que XP no ha inventado ningún método nuevo, sencillamente ha recogido métodos ya existentes y los ha agrupado, y ha comprobado que funcionen. Y para terminar, mencionar que el creador de XP asegura que se garantiza un rato si más no divertido.

2. Historia

La Programación Extrema, como proceso de creación de software diferente al convencional, nace de la mano de Kent Beck (gurú de la XP y autor de los libros más influyentes sobre el tema).

Chrysler Corporation hacía tiempo que estaba desarrollando una aplicación de nóminas, pero sin demasiado éxito por parte de la gente que tenía en el proyecto. El verano de 1996, Beck entró en nómina en la compañía y se le pidió de hacer esta aplicación como trabajo. Es en esta aplicación cuando nace la Programación Extrema como tal.

Beck reconoció que el proceso (o metodología) de creación de software o la carencia de este era la causa de todos los problemas y llegó a la conclusión que para proporcionar un proceso que fuera flexible era necesario realizar ciertos cambios en la estructura o manera de hacer de los programadores, los cuales se tenían que acomodar al cambio a realizar.

El tenía varias ideas de metodologías para la realización de programas que eran cruciales para el buen desarrollo de cualquier sistema.

Las ideas primordiales de su sistema las comunicó en la revista **C++ Magazine** en una entrevista que ésta le hizo el año 1999. En ésta decía que él estaba convencido que la mejor metodología era un proceso que enfatizase la comunicación dentro del equipo, que la implementación fuera sencilla, que el usuario tenía que estar muy informado y implicado y que la toma de decisiones tenía que ser muy rápida y efectiva.

Los autores (o mejor dicho, los propulsores como el propio Kent Beck, Ward Cunningham o Ron Jeffries entre otros) de la Programación Extrema, fueron a la web **Portland Pattern Repository** y empezaron a hablar de ella y promocionarla, de lo que era y cómo realizarla. Estos propulsores de la XP hablaban de ella en cada ocasión que tenían y en cada página que, poco o mucho hablara de temas de programación.

Este hecho, llegó a molestar a buena parte de la comunidad que intentaba discutir sobre temas de programación. Fue tanta esta molestia que nació el fenómeno **XP Free Zone** (zona libre de XP) en determinadas webs como petición de no hablar de Programación Extrema en ella.

La discusión sobre temas de diseño de modelos de programación sobre los cambios recientes se hizo tema difícil porque la mayoría de la actividad fue relacionada con la Programación Extrema. Eventualmente, y debido a ésto, la mayoría de la gente que solía discutir sobre los temas de diseño de modelos de programación fue apartándose de este ambiente para discutir sus ideas en otros ambientes más "reservados". La comunidad empezó a referirse a estos sitios como a **Salas Wiki** (Wards Wiki).

3. Principios básicos

La Programación Extrema se basa en 12 principios básicos agrupados en cuatro categorías:

- **Retroalimentación a escala fina.**
    1. **El principio de pruebas**: se tiene que establecer un período de pruebas de aceptación del programa (llamado también *período de caja negra*) donde se definirán las entradas al sistema y los resultados esperados de estas entradas. Es muy recomendable automatizar estas pruebas para poder hacer varias simulaciones del sistema en funcionamiento. Para hacer estas simulaciones automatizadas, se pueden utilizar *Ambientes de Prueba* (Unit testing frameworks). Un buen ejemplo de un ambiente de prueba es el JUnit para Java (www.junit.org/index.htm). Otros ambientes de pruebas para otros lenguajes como C, C++, JavaScript, XML y servicios Web, pueden encontrarse en www.xprogramming.com/software.htm.
    2. **Proceso de planificación**: en esta fase, el usuario tendrá que escribir sus necesidades, definiendo las actividades que realizará el sistema. Se creará un documento llamado *Historias del usuario* (User Stories). Entre 20 y 80 historias (todo dependiendo de la complejidad del problema) se consideran suficientes para formar el llamado *Plan de Liberación*, el cual define de forma específica los tiempos de entrega de la aplicación para recibir retroalimentación por parte del usuario. Por regla general, cada una de les Historias del usuario suelen necesitar de una a tres semanas de desarrollo.

       Son muy importantes y tienen que ser una constante las reuniones periódicas durante esta fase de planificación. Estas pueden ser a diario, con todo el equipo de desarrollo para identificar problemas, proponer soluciones y señalar aquellos puntos a los que se les ha de dar más importancia por su dificultad o por su punto crítico.
    3. **El cliente en el sitio**: se le dará poder para determinar los requerimientos, definir la funcionalidad, señalar las prioridades y responder las preguntas de los programadores. Esta fuerte interacción cara a cara con el programador disminuye el tiempo de comunicación y la cantidad de documentación, junto con los altos costes de su creación y mantenimiento. Este representante del cliente estará con el equipo de trabajo durante toda la realización del proyecto.
    4. **Programación en parejas**: uno de los principios más radicales y en el que la mayoría de gerentes de desarrollo pone sus dudas. Requiere que todos los programadores XP escriban su código en parejas, compartiendo una sola máquina. De acuerdo con los experimentos, este principio puede producir aplicaciones más buenas, de manera consistente, a iguales o menores costes. Aunque el *pair-programming* puede no ser para todo el mundo, la evidencia anecdótica en la lista de correo de XP (extremeprogramming@yahoogroups.com) demuestra un gran éxito.
- **Proceso continuo en lugar de por lotes.**
    1. **Integración continua**: permite al equipo hacer un rápido progreso implementando las nuevas características del software. En lugar de crear *builds* (o versiones) estables de acuerdo a un cronograma establecido, los equipos de programadores XP pueden reunir su código y reconstruir el sistema varias veces al día. Esto reduce los problemas de integración comunes en proyectos largos y estilo cascada.

2. **Refactorización**: permite a los equipos de programadores XP mejorar el diseño del sistema a través de todo el proceso de desarrollo. Los programadores evalúan continuamente el diseño y recodifican lo necesario. La finalidad es mantener un sistema enfocado a proveer el valor de negocio mediante la minimización del código duplicado y/o ineficiente.
3. **Entregas pequeñas**: colocan un sistema sencillo en producción rápidamente que se actualiza de forma rápida y constante permitiendo que el verdadero valor de negocio del producto sea evaluado en un ambiente real. Estas entregas no pueden pasar las 2 o 3 semanas como máximo.

- **Entendimiento compartido.**

1. **Diseño simple**: se basa en la filosofía de que el mayor valor de negocio es entregado por el programa más sencillo que cumpla los requerimientos. *Simple Design* se enfoca en proporcionar un sistema que cubra las necesidades inmediatas del cliente, ni más ni menos. Este proceso permite eliminar redundancias y rejuvenecer los diseños obsoletos de forma sencilla.
2. Metáfora: desarrollada por los programadores al inicio del proyecto, define una historia de como funciona el sistema completo. XP estimula historias, que son breves descripciones de un trabajo de un sistema en lugar de los tradicionales diagramas y modelos UML (*Unified Modeling Language*). La metáfora expresa la visión evolutiva del proyecto que define el alcance y propósito del sistema.

   Las tarjetas CRC (Clase, Responsabilidad y Colaboración) también ayudarán al equipo a definir actividades durante el diseño del sistema. Cada tarjeta representa una clase en la programación orientada a objetos y define sus responsabilidades (lo que ha de hacer) y las colaboraciones con las otras clases (cómo se comunica con ellas).
3. **Propiedad colectiva del código**: un código con propiedad compartida. Nadie es el propietario de nada, todos son el propietario de todo. Este método difiere en mucho a los métodos tradicionales en los que un simple programador posee un conjunto de código. Los defensores de XP argumentan que mientras haya más gente trabajando en una pieza, menos errores aparecerán.
4. Estándar de codificación: define la propiedad del código compartido así como las reglas para escribir y documentar el código y la comunicación entre diferentes piezas de código desarrolladas por diferentes equipos. Los programadores las han de seguir de tal manera que el código en el sistema se vea como si hubiera estado escrito por una sola persona.

- **Bienestar del programador.**

1. **La semana de 40 horas**: la programación extrema sostiene que los programadores cansados escriben código de menor cualidad. Minimizar las horas extras y mantener los programadores frescos, generará código de mayor calidad. Como dice *Beck*, está bien trabajar tiempos extra cuando es necesario, pero no se ha de hacer durante dos semanas seguidas.

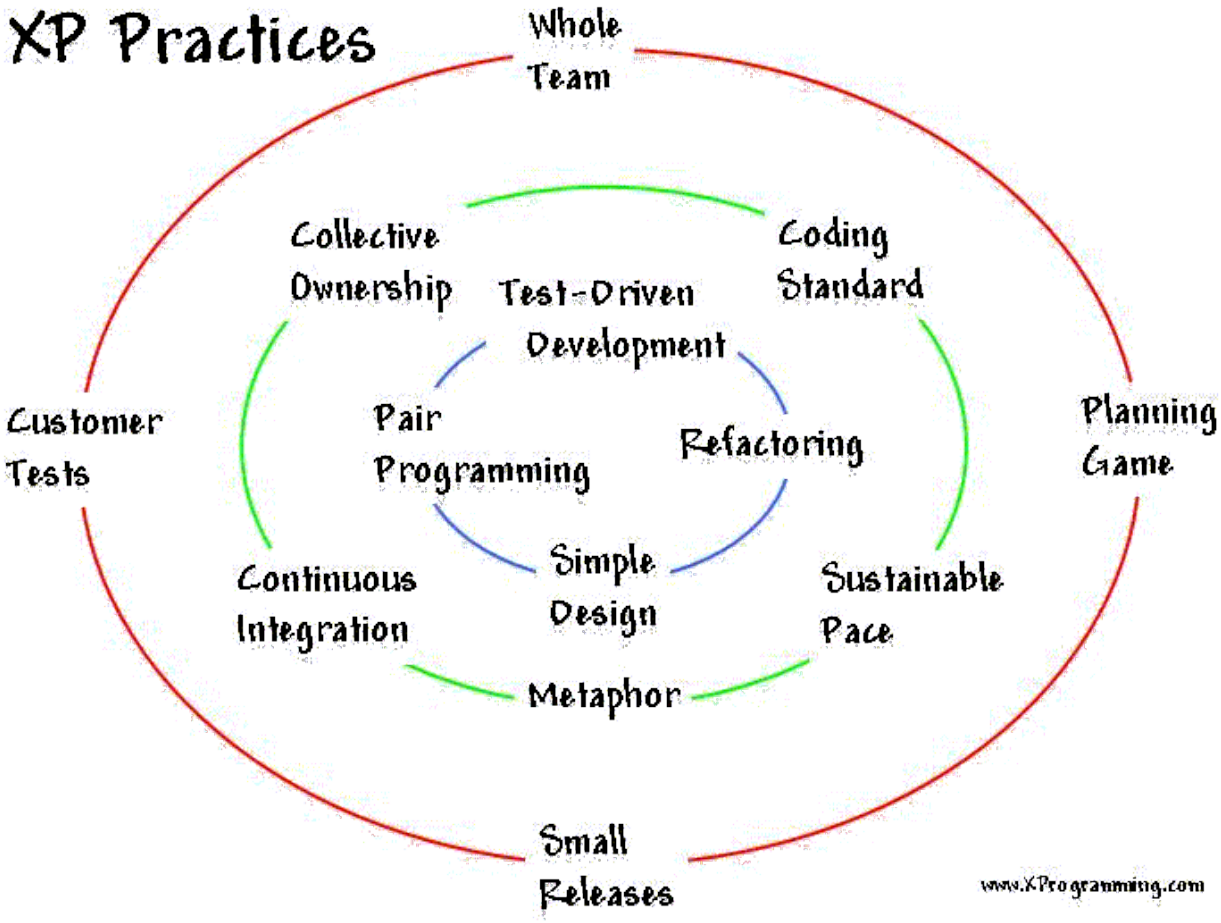


Imagen extraída de la web de Ron Jeffries (www.xprogramming.com)

4. Proceso de desarrollo.

La programación extrema parte del caso habitual de una compañía que desarrolla software, normalmente a medida, en la que hay diferentes roles: un equipo de gestión (o diseño) , uno de desarrollo y los clientes finales. La relación entre el equipo de diseño, los que desarrollan el software y clientes es totalmente diferente al que se ha producido en las metodologías tradicionales, que se basaba en una fase de captura de los requisitos previa al desarrollo, y de una fase de validación posterior al mismo.

4.1. Interacción con el cliente.

En este tipo de programación el cliente pasa a ser parte implicada en el equipo de desarrollo. Su importancia es máxima en el momento de tratar con los usuarios y en efectuar las reuniones de planificación. Tiene un papel importante de interacción con el equipo de programadores, sobre todo después de cada cambio, y de cada posible problema localizado, mostrando las prioridades, expresando sus sensaciones... En este tipo de programación existirán pruebas de aceptación de la programación que ayudarán a que su labor sea lo más provechosa posible.

Al fin y al cabo, el cliente se encuentra mucho más cerca del proceso de desarrollo. Se elimina la fase inicial de recopilación de requerimientos, y se permite que éstos se vayan cogiendo a lo largo del proyecto, de una manera ordenada. De esta forma se posibilita que el cliente pueda ir cambiando de opinión sobre la marcha, pero a cambio han de estar siempre disponibles para solucionar las dudas del equipo de desarrollo.

En XP aparece un nuevo concepto llamado "*Historia de usuario*". Se trata de una lista de características que el cliente necesita que existan en el producto final. Estas constan de dos fases.

- En la primera fase, el cliente describe con sus propias palabras las características y, es el responsable del equipo, el encargado de informarlo de las dificultades técnicas de cada una de ellas y de su coste. A consecuencia de este diálogo, el cliente deja por escrito un conjunto de historias y las ordena en función de la prioridad que tienen pera él. De esta manera ya es posible definir unas fechas aproximadas para ellos.
- En la segunda fase, el cliente cogerá las primeras historias a implementar y las dividirá en trabajos a realizar. El cliente también participa, pero hay más peso por parte del equipo de desarrolladores, esto dará como resultado una planificación más exacta. Este método se repetirá para cada historia.

A diferencia de otras técnicas, como puede ser UML, en el caso de XP, se exige que sea el cliente el encargado de escribir los documentos con las especificaciones de lo que realmente quiere, como un documento de requisitos de usuario.

En esta fase, el equipo técnico será el 'encargado de catalogar las historias del cliente y asignarles una duración. La norma es que cada historia de usuario tiene que poder ser realizable en un espacio entre una y tres semanas de programación. Las que requieran menos tiempo serán agrupadas, y las que necesiten más serán modificadas o divididas.

Finalmente decir que las historias de los usuarios serán escritas en tarjetas, lo que facilitará que el cliente pueda especificar la importancia relativa entre las diferentes historias de usuario, así como el trabajo de los programadores que podrán catalogarlas correctamente. Este formato también es muy útil en el momento de las pruebas de aceptación.

4.2. Planificación del proyecto.

En este punto se tendrá que elaborar la planificación por etapas, donde se aplicarán diferentes iteraciones. Para hacerlo será necesaria la existencia de reglas que se han de seguir por las partes implicadas en el proyecto para que todas las partes tengan voz y se sientan realmente partícipes de la decisión tomada.

Las entregas se tienen que hacer cuanto antes mejor, y con cada iteración, el cliente ha de recibir una nueva versión. Cuanto más tiempo se tarde en introducir una parte esencial, menos tiempo se tendrá para trabajar con ella después. Se aconseja muchas entregas y muy frecuentes. De esta manera un error en la parte inicial del sistema tiene más posibilidades de detectarse rápidamente.

Una de las máximas a aplicar es, los cambios, no han de suponer más horas de programación para el programador, ya que el que no se termina en un día, se deja para el día siguiente.

Se ha de tener asumido que en el proceso de planificación habrán errores, es más, serán comunes, y por esto esta metodología ya los tiene previstos, por lo tanto se establecerán mecanismos de revisión. Cada tres o cinco iteraciones es normal revisar las historias de los usuarios, y renegociar la planificación.

Cada iteración necesita también ser planificada, es lo que se llama *planificación iterativa*, en la que se anotarán las historias de usuarios que se consideren esenciales y las que no han pasado las pruebas de aceptación. Estas planificaciones también se harán en tarjetas, en las que se escribirán los trabajos que durarán entre uno y tres días.

Es por esto que el diseño se puede clasificar como continuo. Añade agilidad al proceso de desarrollo y evita que se mire demasiado hacia delante, desarrollando trabajos que aún no han estado programados.

Este tipo de planificación en iteraciones y el diseño iterativo, hace que aparezca una práctica que no existía en la programación tradicional. Se trata de las discusiones diarias informales, para fomentar la comunicación, y para hacer que los desarrolladores tengan tiempo de hablar de los problemas a los que se enfrentan y de ver como van con sus trabajos.

4.3. Diseño, desarrollo y pruebas.

El desarrollo es la parte más importante en el proceso de la programación extrema. Todos los trabajos tienen como objetivo que se programen lo más rápidamente posible, sin interrupciones y en dirección correcta.

También es muy importante el diseño, y se establecen los mecanismos, para que éste sea revisado y mejorado de manera continuada a lo largo del proyecto, según se van añadiendo funcionalidades al mismo.

La clave del proceso de desarrollar XP es la comunicación. La mayoría de los problemas en los proyectos son por falta de comunicación en el equipo.

En XP, aparece un nuevo concepto llamado Metáfora. Su principal objetivo es mejorar la comunicación entre todos los integrantes del equipo, al crear una visión global y común de lo que se quiere desarrollar. La metáfora tiene que ser expresada en términos conocidos por los integrantes del equipo, por ejemplo comparando el sistema que se desarrollará con alguna cosa de la vida real.

Antes de empezar a codificar se tienen que hacer pruebas unitarias, es decir:

Cada vez que se quiere implementar una parte de código, en XP, se tiene que escribir una prueba

sencilla, y después escribir el código para que la pase. Una vez pasada se amplía y se continúa. En XP hay una máxima que dice "Todo el código que puede fallar tiene que tener una prueba".

Con estas normas se obtiene un código simple y funcional de manera bastante rápida. Por esto es importante pasar las pruebas al 100%.

Respecto a la integración del soft, en XP se ha de hacer una integración continua, es decir, cada vez se tienen que ir integrando pequeños fragmentos de código, para evitar que al finalizar el proyecto se tenga que invertir grandes esfuerzos en la integración final. En todo buen proyecto de XP, tendría que existir una versión al día integrada, de manera que los cambios siempre se realicen en esta última versión.

Otra peculiaridad de XP es que cada programador puede trabajar en cualquier parte del programa. De esta manera se evita que haya partes "propietarias de cada programador". Por esto es tan importante la integración diaria.

Para terminar, otra peculiaridad que tiene la XP. La de fomentar la programación en parejas, es decir, hacer que los programadores no trabajen en solitario, sino que siempre estarán con otra persona. Una pareja de programadores ha de compartir el teclado, el monitor y el ratón. El principio fundamental de este hecho es realizar de manera continua y sin parar el desarrollo de código. Las parejas tienen que ir cambiando de manera periódica, para hacer que el conocimiento se difunda en el grupo. Está demostrado que de esta manera el trabajo es más eficaz y también se consigue más y mejor código.

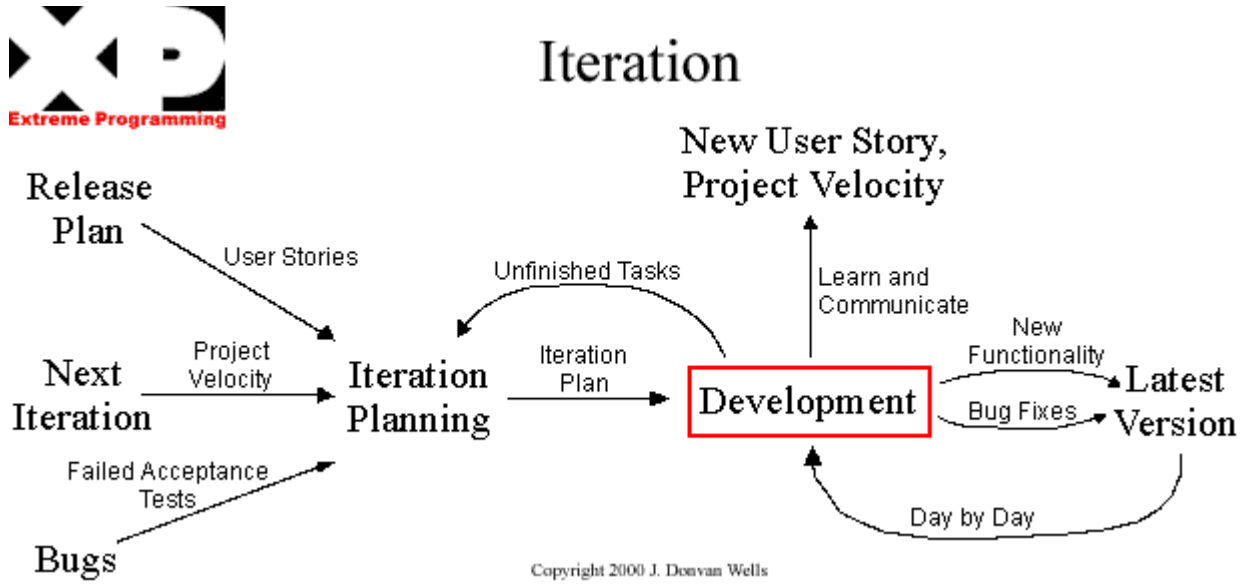


Imagen extraída de www.xprogramming.com

5. Bibliografía

Para aquellos que queráis profundizar más en este tema, aquí os dejo una referencia a libros que hablan de Programación Extrema.

Título: Una explicación de la Programación extrema: aceptar el cambio
Autor: Kent Beck

- **Páginas**: 216 páginas
- **Publicado**: Addison-Wesley Iberoamericana Espanya, S.A. - 2002
- **ISBN**: 8478290559
- **Idioma**: español
- **Nivel**: medio - avanzado

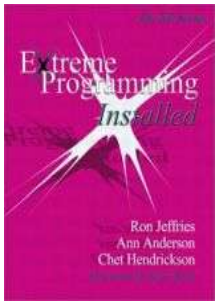
Este libro lo encontrarás en Agapea

Título: La Programación Extrema en la práctica
Autor: James Newkirk, Robert C. Martin

- **Páginas**: 216 páginas
- **Publicado**: Addison-Wesley Iberoamericana Espanya, S.A. - 2002
- **ISBN**: 8478290575
- **Idioma**: español
- **Nivel**: medio - avanzado

Este libro lo encontrarás en Agapea

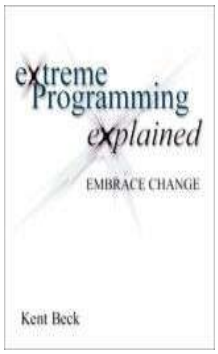
Título: Extreme Programming Installed
Autor: Ron Jeffries, Ann Anderson, Chet Hendrickson, Ronald E. Jeffries



- **Páginas**: 288 páginas
- **Publicado**: Addison-Wesley Pub Co; 1 edición (13 Octubre 2000)
- **ISBN**: 0201708426
- **Idioma**: inglés
- **Nivel**: medio - avanzado

Este libro lo encontrarás en Amazon

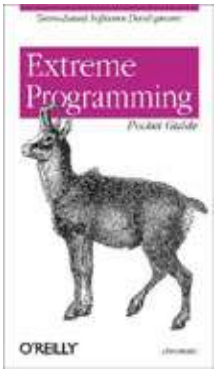
Título: Extreme Programming Explained: Embrace Change
Autor: Kent Beck



- **Páginas**: 224 páginas
- **Publicado**: Addison-Wesley Pub Co; 1 edición (5 Octubre 1999)
- **ISBN**: 0201616416
- **Idioma**: inglés
- **Nivel**: medio - avanzado

Este libro lo encontrarás en Amazon

Título: Extreme Programming Pocket Guide
Autor: chromatic

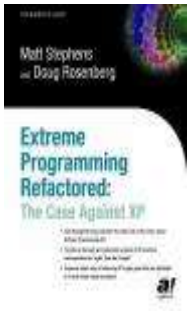


- **Páginas**: 80 páginas
- **Publicado**: O'Reilly & Associates; 1 edición (Junio 2003)
- **ISBN**: 0596004850
- **Idioma**: inglés
- **Nivel**: medio - avanzado

Este libro lo encontrarás en Amazon

Título: Extreme Programming Refactored: The Case Against XP

Autor: Matt Stephens, Doug Rosenberg

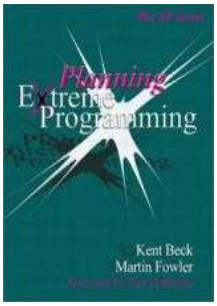


- **Páginas**: 432 páginas
- **Publicado**: APress; (1 Enero 1970)
- **ISBN**: 1590590961
- **Idioma**: inglés
- **Nivel**: medio - avanzado

Este libro lo encontrarás en Amazon

Título: Planning Extreme Programming

Autor: Kent Beck, Martin Fowler



- **Páginas**: 160 páginas
- **Publicado**: Addison-Wesley Pub Co; 2000
- **ISBN**: 0201710919
- **Idioma**: inglés
- **Nivel**: medio - avanzado

Este libro lo encontrarás en Amazon

6. Referencias a webs

Y para acabar el tema de XP, una serie de webs de dónde se ha sacado la información aquí mostrada.

Sólo quería agradecer, antes de terminar, el trabajo de mis compañeros de grupo (este trabajo lo hicimos en una asignatura) Daniel Gomez, Elisabet Aranda y Jordi Fabrega.

web	Extreme Programming: A gentle introduction
Dirección	www.extremeprogramming.org
Enfoque	Divulgativo
Lenguaje	Inglés

web	Extreme Programming: A gentle introduction
Dirección	www.xprogramming.com
Enfoque	Recursos
Lenguaje	Inglés

web	Programación extrema
Dirección	www.programacionextrema.org
Enfoque	Divulgativo
Lenguaje	Español

web	Novatica
Dirección	www.ati.es/novatica
Enfoque	Divulgativo
Lenguaje	Español

web	Java Hispano
Dirección	www.javahispano.org
Enfoque	Divulgativo
Lenguaje	Español

web	DesignUp
Dirección	www.design-up.com/methodes/XP
Enfoque	Divulgativo
Lenguaje	Francés

web	AgileSpain
Dirección	www.agile-spain.com
Enfoque	Divulgativo
Lenguaje	Español