



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Programación extrema

Programación extrema

Contenidos

- Introducción
- Justificación y visión general de XP
- El ciclo de vida en XP
- Los cinco valores básicos
- Las doce prácticas básicas en XP
- Roles de los participantes
- XP Industrial
- Valoración crítica de XP
- Referencias bibliográficas

1. Introducción

La Programación Extrema ***eXtreme Programming (XP)***, es posiblemente la metodología ágil mejor conocida y más estudiada.

Se encarga de llevar ***al extremo*** los principios del desarrollo iterativo:

- Genera ***nuevas versiones*** mejoradas del software ***varias veces al día***.
- El cliente recibe ***cada poco tiempo*** nuevos ***incrementos funcionales*** del software.
- Se debe ejecutar todas las pruebas en cada nueva versión, y sólo se acepta la versión si todas las pruebas se ejecutan correctamente.

Aunque muchos de sus principios y prácticas comenzaron a desarrollarse en la década de 1980, todas las ideas que subyacen en esta metodología han venido siendo formalizadas en los trabajos de Kent Beck, a quien podemos considerar el padre de XP (Beck, 1999, 2000).

2. Justificación y visión general de XP

El primer proyecto desarrollado con XP comenzó el 6 de marzo de 1996 (Wells, 2013), y desde entonces ha demostrado su utilidad en infinidad proyectos en una gran variedad de industrias y tamaños de equipos de desarrollo.

La principal razón de su éxito radica en el énfasis que hace en la satisfacción del cliente y el trabajo en equipo.

En el equipo de trabajo aparecen integrados todos los roles principales (managers, clientes y desarrolladores) en un proceso de comunicación de igual a igual, que se realiza preferiblemente siempre de manera oral.

2. Justificación y visión general de XP

En el modelo tradicional en cascada, el proceso comienza con la interacción con clientes y usuarios, obteniendo información sobre las características del sistema deseado.

A continuación, se diseña el sistema, se implementa el código, y finalmente se realizan las pruebas para garantizar que el sistema entregado es el correcto.

Esto presenta en ocasiones algunos inconvenientes:

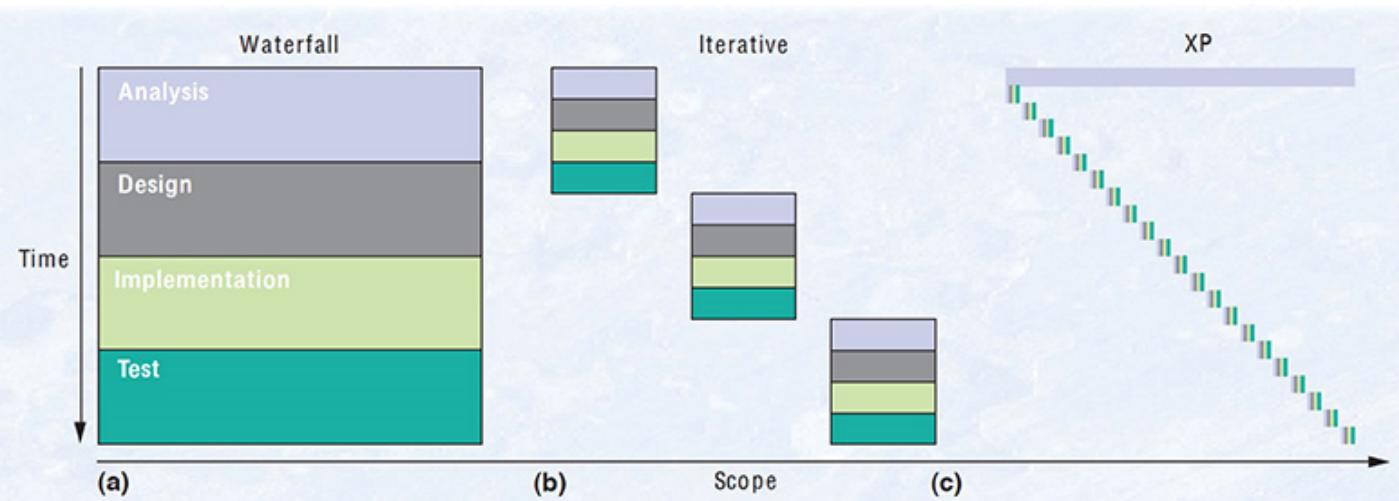
- › el usuario no conoce o no sabe expresar sus necesidades
- › o cambia de opinión
- › o simplemente desde el equipo no se entienden bien estas necesidades

de manera que el sistema final no es el correcto.

cuando los problemas se
detectan tarde puede
resultar técnica °
económicamente difícil
solucionarlos.

2. Justificación y visión general de XP

Una manera de mejorar el proceso consiste en la adopción de **un modelo iterativo** que descompone la duración total del proyecto en varios ciclos consecutivos, respetando las mismas fases en cada uno de ellos.



2. Justificación y visión general de XP

Este modelo permite detectar con mayor antelación posibles errores, y resulta más flexible y adaptable al cambio.

Se plantea un modelo con iteraciones muy cortas, con fases de diseño, implementación y pruebas solapadas entre sí, y dedicando a las labores de análisis en cada iteración el tiempo estrictamente necesario para resolver los problemas correspondientes a ese incremento concreto.

Se evita así el planteamiento de un diseño monolítico completo inicial del sistema, puesto que se asume que posibles cambios pueden hacer que este esfuerzo resulte inútil.

Con la Programación extrema se lleva al límite este principio reduciendo al máximo la duración de las iteraciones.

2. Justificación y visión general de XP

Cabe destacar que XP considera dedicar un tiempo a **labores de análisis** antes del comienzo del proyecto, definiendo de la manera más precisa posible el funcionamiento esperado del sistema.

El resultado es una especificación inicial del sistema materializada en **un conjunto de historias de usuario** que redacta el cliente.

A partir de ahí, arranca el proyecto que se descompone en **iteraciones**, la implementación de cada una de las tareas, y las correspondientes entregas (releases).

Ken Beck propone una duración mínima de un mes para esta tarea – suponiendo un proyecto de 10 personas de un año de duración (Beck, 1999)–.

2. Justificación y visión general de XP

El flujo general de un proyecto en XP:

- El cliente elige el subconjunto más pequeño posible de historias de usuario, con coherencia interna y priorizadas en función del valor que aportan sobre el producto final, y ***define el alcance de cada release.***
- Para estimar la cantidad de historias que es posible implementar, dialoga con el equipo de desarrollo en lo que se conoce como el ***«juego de la planificación»*** (mecanismo fundamental de planificación en XP).
- Cada release se divide en una o más iteraciones. En cada iteración se escoge un grupo de historias asociadas al release, y cada una de ellas se ***descompone en tareas*** que se asignan los miembros del equipo.
- La implementación de las tareas comienza con la definición de ***las pruebas unitarias***, que es posiblemente la práctica más importante que se encuentra en el corazón de XP.

2. Justificación y visión general de XP

El flujo general de un proyecto en XP:

- La definición de las pruebas, el diseño del incremento y la codificación hasta que todas las pruebas sean superadas son operaciones que se realizan siempre en *parejas de programadores*.
- En paralelo, el cliente está encargado de definir las *pruebas de aceptación* que validarán la superación de la iteración considerada, y de la release en su conjunto.

3. El ciclo de vida en XP

La programación extrema propone *cuatro actividades estructurales* que definen el proceso de desarrollo:

- ➊ Planificación
- ➋ Diseño
- ➌ Codificación
- ➍ Pruebas

Además, se sigue un enfoque eminentemente *orientado a objetos*, aspecto que se tiene en cuenta en las labores de diseño y codificación.

Un proceso inicial y necesario es la *captura de requisitos*, que se concreta formalmente en *historias de usuario (User Stories)*.

3. El ciclo de vida en XP

Las historias de usuario:

- Capturan los requisitos ***definidos por el cliente*** y contienen funcionalidades que aportan valor al producto desde el punto de vista del negocio y de los usuarios finales.
- Son ***creadas por el cliente*** y sirven como base para la definición de las pruebas de aceptación del sistema.
- El tiempo de desarrollo suele ser de **una a tres** semanas y si resulta superior es síntoma de que la historia debe descomponerse.
- Las escribe el propio cliente, ***evitando el empleo de lenguaje técnico***, y generalmente se escribe cada una en una tarjeta numerada, de manera que cada historia queda identificada.

Su propósito es similar al de los casos de uso, pero están más orientadas a realizar estimaciones de complejidad y tiempo de implementación durante la fase de planificación de la release (Wells, 1999).

3. El ciclo de vida en XP

Ejemplo de una historia de usuario:

Título: Creación de nueva cita	ID: 27
Historia de usuario: <i>Como recepcionista de la clínica quiero poder insertar nuevas citas médicas en el sistema para poder organizar la actividad de consultas del centro médico.</i>	
Prioridad: Alta	Estimación (días): 6

3. El ciclo de vida en XP

Ejemplo de una historia de usuario - información adicional:

Comentarios:

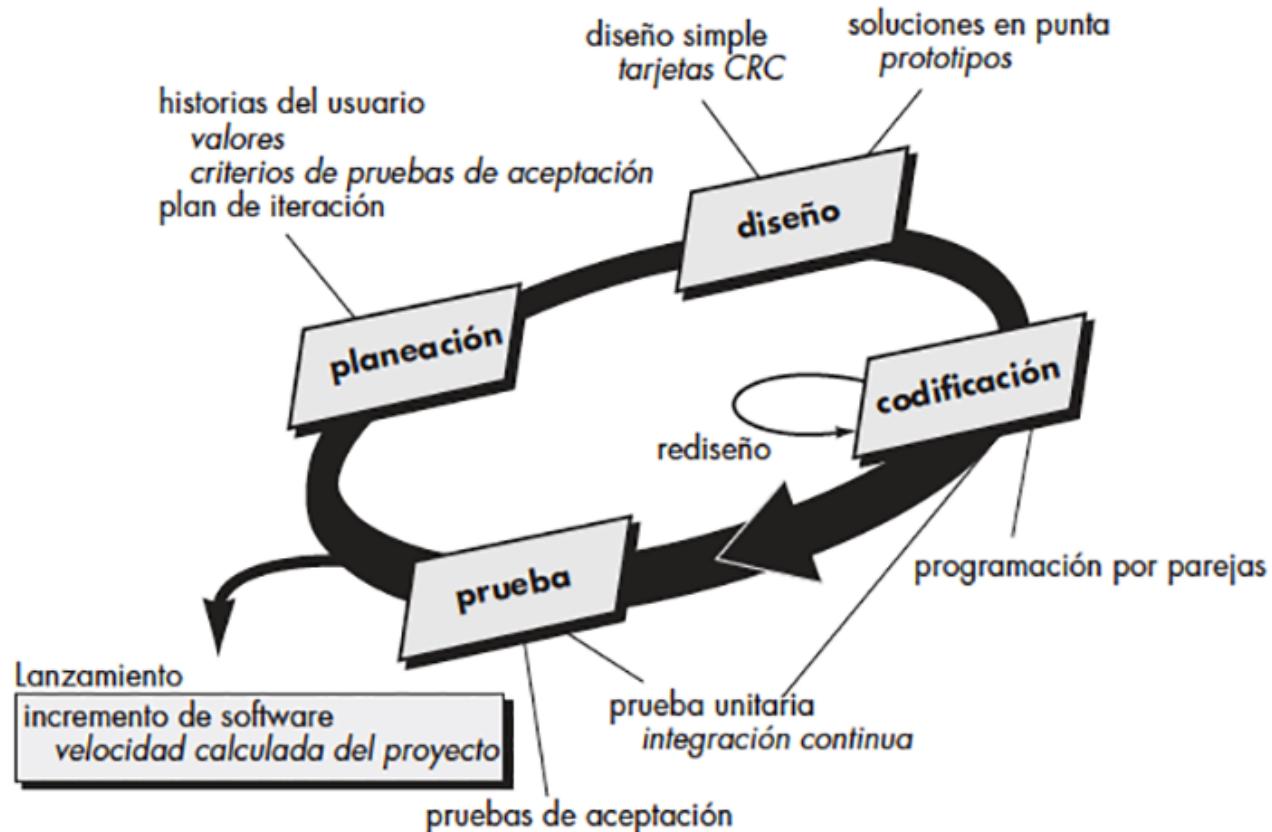
- Los datos obligatorios de cada cita son: el nombre del paciente, la fecha y la hora, el nombre del médico asignado y una descripción breve del motivo de la consulta.
- Para poder insertar nuevas citas se debe disponer de una interfaz que muestre un calendario y represente de manera gráfica los horarios disponibles.

Criterios de aceptación:

- No se puede insertar una cita sin indicar todos los datos obligatorios.
- La información de la nueva cita queda almacenada en la base de datos.

3. El ciclo de vida en XP

Esquema general del modelo de proceso en XP:



3. El ciclo de vida en XP

Fase de planificación:

Se trata de planificar la siguiente entrega (release) y la siguiente iteración dentro de cada release.

El cliente prioriza aquellas historias que aportan mayor valor al negocio desde el punto de vista del cliente y negocia con el equipo de desarrollo su implementación en lo que se conoce como el **juego de la planificación**.

En este punto, **los programadores analizan las historias** y las **descomponen en tareas individuales** que son necesarias para su implementación.

Una vez definidas las tareas, **se asignan a los diferentes programadores** que, a partir de este momento, **trabajarán en parejas** definiendo las pruebas unitarias, diseñando la solución, implementando e integrando el nuevo código.

Finalmente, podemos considerar que en XP **las tareas de planificación** constituyen una herramienta **temporal y efímera, adaptable** y que se realizan en los tres niveles fundamentales: **Release, Iteración y Tareas**.

3. El ciclo de vida en XP

Niveles de planificación en XP:

Nivel	Tiempo	Entradas	Salidas
Release	Meses	Todas las historias restantes	Historias por implementar en el release
Iteración	Semanas	Historias restantes del release	Historias por implementar en la iteración
Tareas	Días - horas	Historias por implementar en la iteración	tareas asignadas a los desarrolladores

3. El ciclo de vida en XP

Descomposición de una historia de usuario en tareas:

Una posible descomposición de la historia representada anteriormente podría ser la siguiente:

En este caso, la suma de horas correspondientes a estas tres tareas no coincide con la estimación inicial para la historia en su conjunto.

Construir el modelo de datos que representa una cita individual + actualizar la base de datos (1 día).

Diseñar la interfaz de usuario para inserción de nuevas citas (8 días).

Esto es algo comprensible, pues en este caso estamos realizando un análisis mucho más fino y, además, la previsión se realiza teniendo en cuenta las condiciones actuales del proyecto, en cuanto a grado de avance y recursos disponibles.

Escribir la lógica del controlador que relaciona la interacción del usuario con el calendario y la modificación del modelo (2 días).

3. El ciclo de vida en XP

Descomposición de una historia de usuario en tareas:

Para solucionar esta discrepancia encontramos dos alternativas:

- ***Adelantar el análisis*** y la descomposición en tareas en el momento en que se define cada historia de usuario, de manera que podamos asignar una estimación más precisa en la tarjeta. Este enfoque tiene dos problemas: exige adelantar actividades que pueden no ser necesarias (por ejemplo, porque la historia es modificada o simplemente desaparece) y, además, no considera posibles dependencias de implementación con otras tareas, que pueden estar más claras en la iteración concreta.
- ***Utilizar puntos de historia*** en las tarjetas de historia de usuario. Esta es otra manera de estimar la complejidad de implementación, perfectamente válida, que simplemente pone en relación unas historias con otras respecto de la previsión de la dificultad de su implementación. Cumple en el objetivo de comparar las historias entre sí, pero evita la necesidad de ofrecer una estimación temporal precisa.

3. El ciclo de vida en XP

Diseño

En la fase de diseño, eminentemente orientada a objetos, es habitual la utilización de **tarjetas CRC** (Clase-Responsabilidad-Colaborador).

Ejemplo de CRC de la historia de usuario «creación de nueva cita».

Nombre de la clase: <u>Appointment</u>	
Responsabilidades	Colaboradores
<ul style="list-style-type: none">▪ Inserta nuevas citas de pacientes.▪ Recupera la información de una cita a partir del ID.▪ Recupera todas las citas asociadas a un paciente.▪ Recupera todas las citas asociadas a un médico.▪ Recupera todas las citas asociadas a una fecha concreta.	<ul style="list-style-type: none">▪ <u>Doctor</u>▪ <u>Patient</u>

La fase de diseño no es simplemente la primera en cada iteración, sino que **las labores de diseño se realizan continuamente**, incluso durante el proceso de codificación, aplicando así las prácticas recomendadas de mantener el diseño lo más sencillo posible y la refactorización

3. El ciclo de vida en XP

Codificación

El verdadero punto de arranque de la implementación en cada iteración es ***la definición de las pruebas unitarias.***

La propia definición de las pruebas ayuda a los desarrolladores a ***comprender mejor la tarea,*** a analizarla en profundidad y a plantear cualquier posible duda al cliente *que está integrado en el propio equipo de desarrollo.*

La culminación de la implementación de la tarea será precisamente la comprobación de que las pruebas asociadas se ejecutan satisfactoriamente.

Todo el proceso de desarrollo aplica la práctica de ***programación en parejas,*** y a medida que las tareas se finalizan, el código se integra en el repositorio general, ejecutándose todas las pruebas para comprobar que las modificaciones no han introducido ningún tipo de problema o error.

3. El ciclo de vida en XP

Pruebas

Las pruebas unitarias son el elemento fundamental que ***guía y valida todo el proceso de desarrollo.***

Estas pruebas permiten comprobar la correcta implementación de cada una de las tareas y, por tanto, dan por finalizada la labor de los desarrolladores.

```
# Prueba para verificar el valor total de una mano
# Resultado: 21 (True)
def test_get_hand_total():
    dealer.hand = ['A', 'J']
    assert dealer.get_hand_total() == 21

# Prueba para verificar el valor total de una mano
# Resultado: 18 (True)
def test_get_hand_total_noases():
    dealer.hand = ['3', 'Q', '5']
    assert dealer.get_hand_total() == 18
```

3. El ciclo de vida en XP

Pruebas

Se ejecutan en último momento *las pruebas de aceptación*, definidas por el cliente, que *validan que la historia de usuario en su conjunto ha sido correctamente implementada*.

Feature: The dealer for the game of 21

Scenario: Deal initial cards

Given a dealer

When the round starts

Then the dealer gives itself two cards

```
@given('a dealer')
def step_impl(context):
    context.dealer = Dealer()

@when('the round starts')
def step_impl(context):
    context.dealer.new_round()

@then('the dealer gives itself two cards')
def step_impl(context):
    assert (len(context.dealer.hand) == 2)
```

4. Los cinco valores básicos

Kent Beck establece cinco valores básicos que guían todas las actividades dentro de un proyecto desarrollado bajo XP (Beck, 2000):



5. Las doce prácticas básicas en XP

La programación extrema propone una serie de prácticas de programación que tienen en cuenta los valores en los que se apoya la metodología, e intentan plasmarlos en técnicas concretas.

- › El juego de la planificación
- › Las pequeñas entregas
- › La metáfora del sistema
- › El diseño simple
- › Test-Driven Development
- › La refactorización
- › La programación en parejas
- › La propiedad del código compartida
- › La integración continua
- › La semana de 40 horas
- › La involucración del cliente
- › Empleo de estándares de programación

5. Las doce prácticas básicas en XP

El juego de la planificación

Se trata de determinar de manera rápida el alcance de cada iteración combinando las prioridades de negocio con la factibilidad técnica en un proceso dialogado.

- En cada iteración, es el cliente quien decide cuál será el alcance de la implementación y fija los plazos, basándose en las estimaciones realizadas por el equipo de desarrollo.
- Los programadores, por su parte, implementan sólo aquellas funcionalidades definidas en las historias incluidas en la iteración.

5. Las doce prácticas básicas en XP

Decisiones de negocio	Decisiones técnicas
<ul style="list-style-type: none">▶ Alcance: se trata de maximizar el valor aportado en la iteración.▶ Prioridad: se trata de establecer preferencias relativas entre las características que se deben implementar.▶ Fechas de las entregas: valorando en qué momentos en el tiempo es necesaria la liberación de nuevas versiones.	<ul style="list-style-type: none">▶ Estimaciones de la duración de implementación de las características.▶ Consecuencias de las decisiones técnicas adoptadas.▶ Proceso de desarrollo y organización interna del equipo.▶ Planificación detallada del orden de implementación de historias y tareas dentro de cada <i>release</i>.

5. Las doce prácticas básicas en XP

Las pequeñas entregas

- Se busca poner el sistema en producción cuanto antes, priorizando aquellas funcionalidades que aportan más valor.
- A partir de ese momento, se continúa mejorando el sistema, introduciendo cada vez un pequeño conjunto de mejoras en iteraciones muy cortas.
- Se debe tener en cuenta que la nueva release debe tener sentido, constituir una aportación significativa y no debe implementar funcionalidades a medias, aunque con ello se consiga acortar el ciclo de desarrollo.

5. Las doce prácticas básicas en XP

La metáfora del sistema

- Se trata de guiar todo el proceso de desarrollo proporcionando una historia o descripción sencilla que captura el funcionamiento general del sistema, sus objetivos y ambición, de manera que todos los participantes puedan comprenderla.
- Una famosa metáfora es la del «escritorio» de los modernos sistemas operativos con interfaz gráfica de usuario.

La metáfora del escritorio representa lo que, desde un enfoque más técnico, consideraríamos la arquitectura general del sistema, destacando los elementos fundamentales del producto al más alto nivel, pero omitiendo el vocabulario técnico, de manera que resulte comprensible por participantes con formación técnica o sin ella.

5. Las doce prácticas básicas en XP

La metáfora del sistema

La metáfora es una herramienta poderosa para explicar la funcionalidad general de un sistema e incluso las principales características de su arquitectura, de manera que resulte fácilmente comprensible para cualquiera.

Un ejemplo típico es el de un **servicio de atención al cliente** (*call center*). En este caso, la solución de las necesidades de un cliente puede requerir dar un número determinado de pasos, hablando con diferentes asistentes y técnicos, humanos o virtuales. En este caso, el símil puede establecerse con una **cadena de montaje física**, donde existen distintas estaciones de trabajo, con técnicos especializados en resolver problemas concretos.

5. Las doce prácticas básicas en XP

El diseño simple

- El sistema debe mantener un diseño tan simplificado como sea posible en un momento dado y cualquier complejidad adicional debe eliminarse en el momento en que sea descubierta.
- El diseño correcto para el sistema en un momento dado es aquel que (Beck, 2000):
 - *Permite ejecutar todos los test de manera satisfactoria.*
 - *No contiene lógica duplicada.*
 - *Contiene el menor número posible de clases y métodos.*

5. Las doce prácticas básicas en XP

Test-Driven Development (desarrollo dirigido por pruebas)

- Los programadores comienzan definiendo las pruebas unitarias del sistema, proceso que les permite comprender mejor los requisitos y sirve como hilo conductor del proceso de desarrollo.
- El cliente está encargado de construir las pruebas funcionales que servirán para validar que las historias han sido correctamente implementadas.

En el contexto de XP,
cualquier característica
que no tenga al menos una
prueba automatizada
asociada, simplemente no
existe.

5. Las doce prácticas básicas en XP

Test-Driven Development (desarrollo dirigido por pruebas)

- Esta definición de pruebas inicial aumenta la confianza que, tanto clientes como desarrolladores, tienen en el código, y al mismo tiempo aumenta la seguridad para afrontar posibles cambios imprevistos.
- Si la introducción del cambio causa algún problema en características previas, es posible detectarlo de manera prematura.

5. Las doce prácticas básicas en XP

La refactorización

- Es labor rutinaria de los programadores la reestructuración del sistema respetando la funcionalidad implementada, buscando siempre la simplicidad y flexibilidad y, por tanto, facilitando la comunicación y la retroalimentación de las características del sistema y su diseño.
- Al mismo tiempo que se introducen nuevas características se reflexiona sobre la posibilidad de reestructurar el programa de manera que la ampliación resulte sencilla y, una vez implementada la funcionalidad, se busca el refinamiento y simplificación del código.

5. Las doce prácticas básicas en XP

La refactorización

- La refactorización del software es una técnica de ingeniería que trata de mejorar la estructura del código respetando su funcionalidad.
- Martin Fowler ejemplos de refactorización:

extracción de un método:

código origen	resultado
<pre>5 void printOwing() { 6 printBanner(); 7 8 //print details 9 System.out.println ("name: " + _name); 10 System.out.println ("amount " + getOutstanding()); 11 }</pre>	<pre>14 void printOwing() { 15 printBanner(); 16 printDetails(getOutstanding()); 17 } 18 19 void printDetails (double outstanding) { 20 System.out.println ("name: " + _name); 21 System.out.println ("amount " + outstanding); 22 }</pre>

5. Las doce prácticas básicas en XP

La refactorización

método inline:

código origen	resultado
<pre>32 int getRating() { 33 return (moreThanFiveLateDeliveries()) ? 2 : 1; 34 } 35 boolean moreThanFiveLateDeliveries() { 36 return _numberOfLateDeliveries > 5; 37 }</pre>	<pre>42 int getRating() { 43 return (_numberOfLateDeliveries > 5) ? 2 : 1; 44 }</pre>

5. Las doce prácticas básicas en XP

La programación en parejas

- Idealmente, cualquier actividad de implementación de código la realizan dos programadores sentados en la misma máquina (pair programming).
- Mientras que uno de ellos codifica o diseña, el otro supervisa.
- El programador que escribe el código piensa sobre la mejor implementación mientras escribe, mientras que el otro puede adoptar una postura más estratégica, reflexionando sobre las posibles implicaciones de la modificación o alternativas de mejora.
- Los emparejamientos se modifican frecuentemente, de manera que esta práctica también facilita la comunicación, y consigue que todos los miembros del equipo estén al tanto de todas las características del sistema.

5. Las doce prácticas básicas en XP

La propiedad del código compartida

- Cualquier miembro del equipo de desarrollo tiene la potestad y autoridad para modificar cualquier parte del código que estime oportuno, si en ello ve una oportunidad de mejora.
- De esta manera, todo el equipo es responsable del sistema en su conjunto. Esto tiene al menos dos implicaciones:
 - *Todos los miembros del equipo deben tener acceso a todas las partes del código. Para ello es necesaria la utilización de algún sistema de control de versiones, uso de git.*
 - *Todos los miembros del equipo tienen una visión general del código producido, todos tienen algún nivel de conocimiento sobre cada componente.*

5. Las doce prácticas básicas en XP

La integración continua

- Se debe disponer de la configuración apropiada para facilitar la integración y construcción del sistema (incluida la ejecución de las pruebas) tantas veces como sea necesario y, al menos, cada vez que una tarea sea finalizada.
- De esta manera, todo el nuevo código con sus modificaciones se integra y verifica cada pocas horas (al menos cada día) e, idealmente, se integra cada vez que un programador (o pareja de programadores) finaliza una tarea.
- De esta manera es más obvio en quién recae la responsabilidad del problema, en caso de que algún test quede roto.

5. Las doce prácticas básicas en XP

La semana de 40 horas

- Una regla fundamental es no trabajar cada semana más de 40 horas y, por supuesto, no hacer horas extras o, al menos, no hacerlas durante dos semanas seguidas.
- Así, vemos cómo, desde el punto de vista teórico, XP se preocupa por el bienestar del programador (nombre que en ocasiones recibe esta práctica).
- Beck (2000) hace referencia en su libro a la diferencia de culturas entre las empresas europeas y americanas. En el viejo continente es habitual que los empleados tomen dos o tres semanas seguidas de vacaciones y en las empresas americanas rara vez se descansa más de algunos pocos días seguidos. ***Beck prefiere la práctica europea.***

5. Las doce prácticas básicas en XP

La involucración del cliente

- Una práctica muy recomendada es la inclusión, dentro del equipo, del cliente y, si es posible, de un usuario (**cliente in situ**).
- Se trata de que esté disponible en cualquier momento para resolver cualquier tipo de cuestión o incertidumbre.
- Esta práctica tiene el inconveniente de que el usuario involucrado desatiende sus ocupaciones habituales para atender al equipo respondiendo dudas cotidianas.
- Con ello se persigue una mejor calidad del resultado final y la rápida resolución de cualquier tipo de incertidumbre por parte del equipo técnico.

5. Las doce prácticas básicas en XP

Empleo de estándares de programación

- Es importante que todo el proceso de codificación se ajuste a estándares, dependientes del lenguaje y plataforma utilizados, lo cual redunda en una mejor compresión y mantenibilidad del código.
- Esta práctica se vuelve especialmente necesaria en el contexto de la aplicación de la programación en parejas (donde los programadores van rotando y modificando diferentes partes del código) y teniendo en cuenta la propiedad del código compartida.
- Podría resultar inmanejable que cada uno de los desarrolladores aplique sus propios convenios y prácticas de programación concretas.

6. Roles de los participantes

La programación extrema propone una serie de roles, o perfiles específicos, que deben participar en el proyecto de desarrollo.

La idea es que los equipos funcionan mejor cuando se definen perfiles concretos que aceptan la responsabilidad de tareas determinadas, consiguiendo una especialización que redunde en el beneficio de todos.

Programador (programmer)

Es el centro de XP. En el contexto de los valores y prácticas que aquí se proponen, el programador adquiere un rol más relevante y específico que en otras metodologías.

Además de las tareas habituales de programación, refactorización, definición de pruebas, etcétera, se espera que el programador posea (Beck, 2000):

6. Roles de los participantes

CUALIDADES DEL PROGRAMADOR	
Habilidades comunicativas	Facilitando la comunicación con el cliente, o la programación en parejas.
Hábito de la simplicidad	Comprometiéndose a producir lo estrictamente necesario para complacer al cliente, y manteniendo un diseño sencillo.
Generosidad y respeto	Admitiendo la propiedad compartida del código y el hecho de que otros miembros del equipo puedan modificar el código escrito por uno.
Coraje	Para afrontar los propios miedos a parecer poco inteligente o no lo suficientemente bueno, problemas a los que se enfrenta cualquier desarrollador en algún momento de su vida.

6. Roles de los participantes

El cliente (customer)

Una característica fundamental de la metodología XP es que el cliente es considerado un miembro más del equipo.

El programador sabe **cómo programar**, mientras que el cliente o usuario sabe **qué se debe programar** (Beck, 2000).

- Es el encargado de tomar las decisiones más estrechamente vinculadas con la visión estratégica y de negocio del producto.
- Escribir buenas historias de usuario.
- Está encargado de definir las pruebas de aceptación que se ejecutan al final de cada iteración.

6. Roles de los participantes

Encargado de pruebas (tester)

- Está encargado de ayudar al cliente con las pruebas de aceptación, sobre todo en aquellos casos en los que este último carece de los conocimientos técnicos.
- Define las pruebas unitarias que guían el proceso de desarrollo en cada iteración.
- No necesariamente es una figura independiente de la del programador, pero sí es necesario que alguien acepte la responsabilidad de controlar que todos los test están definidos y se ejecutan correctamente en cada integración.

6. Roles de los participantes

Rastreador (tracker)

Su labor está estrechamente relacionada con la estimación de esfuerzo de las tareas, que está eminentemente basada en la experiencia. El rastreador analiza las estimaciones realizadas por los miembros del equipo, y detecta posibles desviaciones sistemáticas.

El metric man observa sin molestar y almacena datos históricos del proceso.

También tiene una visión general del proceso de desarrollo, anotando información sobre los resultados históricos de las pruebas, el número de defectos detectados o el progreso general de cada iteración, alertando de posibles retrasos.

6. Roles de los participantes

Entrenador (coach)

Es responsable del proceso en su conjunto y tiende a mantenerse en un plano más secundario a medida que el equipo madura y sus funciones van perdiendo relevancia.

Especialmente al comienzo de un proyecto XP, con un equipo poco experimentado, debe asegurarse de que todos los participantes conocen su papel y saben trabajar con las prácticas adoptadas.

También debe ser capaz de introducir nuevas prácticas que considere pertinentes, detectar el estado de ánimo del equipo, y sugerir acciones de mejora en cuanto a diseño, implementación y pruebas.

6. Roles de los participantes

Jefe de proyecto (big boss o manager)

Es el garante de que el proyecto finaliza cumpliendo los objetivos. Su principal valor es el coraje, pues se espera de él que en momentos críticos tome decisiones importantes.

Está encargado de organizar y moderar las reuniones y asegurar las condiciones óptimas para el desarrollo del proyecto.

Su función está relacionada con el aseguramiento de que el desarrollo del proyecto se ajusta a lo planificado, pero no tanto con organizar el trabajo *definiendo qué tiene que hacer cada participante* pues esto se decide directamente a través del diálogo entre cliente y programador.

7. XP Industrial

XP Industrial ***Industrial XP (IXP)*** surgió como respuesta a la necesidad de escalar XP en organizaciones con más de 500 empleados (Kerievsky, 2005).

Intenta mantener el espíritu minimalista, con pequeñas iteraciones y gran importancia de las prácticas de ingeniería, como propone XP tradicional, pero haciendo un esfuerzo en la mayor inclusión de la gerencia y la participación del cliente.

Introduce seis nuevas prácticas y actualiza algunas de las tradicionales:

- ➊ Evaluación de factibilidad.
- ➋ Comunidad del proyecto.
- ➌ Calificación del proyecto.
- ➍ Gestión orientada a pruebas.
- ➎ Retrospectivas.
- ➏ Aprendizaje continuo.

7. XP Industrial

Nuevas prácticas

Evaluación de factibilidad:

- Se trata de una actividad que realiza un experto en IXP para asegurar que la organización está preparada para aplicar esta metodología.
- Durante uno o dos días el experto se entrevista con posibles participantes y analiza si están preparados para acometer el desarrollo, tanto desde el punto de vista técnico, como desde el punto de vista de la interiorización de los valores de XP.

7. XP Industrial

Nuevas prácticas

Comunidad del proyecto:

- Se extiende la idea de equipo a la idea de comunidad, que incluye, además del equipo técnico y al menos un miembro del lado del cliente, otros agentes (del equipo jurídico, de ventas y marketing, etc.).
- En una comunidad encontramos participantes más o menos involucrados, en función de sus responsabilidades y nivel de conocimientos.
- En este caso, al aumentar el número de participantes, es necesario definir previamente los papeles de cada uno y los canales de coordinación y comunicación.

7. XP Industrial

Nuevas prácticas

Calificación del proyecto:

- En esta actividad se trata de analizar el proyecto minuciosamente y contrastarlo con los objetivos generales de la organización para valorar su adecuación, necesidad y pertinencia.
- Entre otros aspectos, se analiza la comunidad del proyecto, *la visión* (futuro que deseamos alcanzar con el desarrollo) y *la misión* (la estrategia para alcanzar la visión).

7. XP Industrial

Nuevas prácticas

Gestión orientada a pruebas:

- Se trata de establecer métricas y pruebas que permitan validar en qué medida el proyecto es un éxito o un fracaso, más allá del aspecto puramente técnico, incluyendo objetivos de la organización más amplios.
- Las pruebas de gestión son afirmaciones que indican un objetivo medible, acotado en el tiempo y con un resultado binario (se alcanza o no se alcanza el objetivo).
- Las buenas pruebas de gestión son SMART: **específicas** (specific), **medibles** (measurable), **alcanzables** (achievable), **relevantes** (relevant) y **temporalizables** (time-based).

7. XP Industrial

Nuevas prácticas

- Ejemplo de prueba de gestión
 - *El enunciado de una prueba de gestión para una compañía como Apple y su servicio iTunes podría ser el siguiente (Industrial Logic, 2004):*

Nuestro nuevo servicio registrará al menos un millón de descargas de canciones durante el primer mes en producción.

Si durante la primera semana se consigue alcanzar este objetivo, entonces se considera que el test ha sido superado.

7. XP Industrial

Nuevas prácticas

Retrospectivas:

- Están orientadas a la mejora del propio proceso IXP y consisten en el análisis retrospectivo, después de cada entrega de un incremento, que permite aprender a partir del transcurso de la iteración.

7. XP Industrial

Nuevas prácticas

Aprendizaje continuo:

- Se incentiva que los participantes del proyecto aprendan nuevas herramientas y técnicas que puedan repercutir en la mejora del proceso global.
- Es una actividad de mejora más centrada en el nivel individual y el aprendizaje de conocimientos tanto técnicos como no técnicos.
- Una práctica tradicional que facilita este aprendizaje dentro del grupo es la programación en parejas.
- En el caso de IXP, las comunidades suelen tener sesiones de estudio en grupo cada una o dos semanas

7. XP Industrial

Mejora de prácticas existentes

IXP introduce algunas modificaciones sobre las prácticas tradicionales de XP, que en su mayoría están orientadas a incluir de manera más clara a clientes y gestores en la práctica diaria.

SDD (Storytest–Driven Development), o desarrollo dirigido por historias:

- Es una mejora sobre la práctica TDD que trata de no comenzar a escribir código hasta que haya definida una prueba para cada una de las historias de usuario.
- Son las pruebas de aceptación del sistema.

DDD (Domain–Driven Design), o diseño dirigido a partir del dominio:

- Es una mejora sobre la metáfora del sistema, que busca formalizar el conocimiento que un experto tiene sobre el dominio de aplicación.

7. XP Industrial

Mejora de prácticas existentes

Emparejamientos:

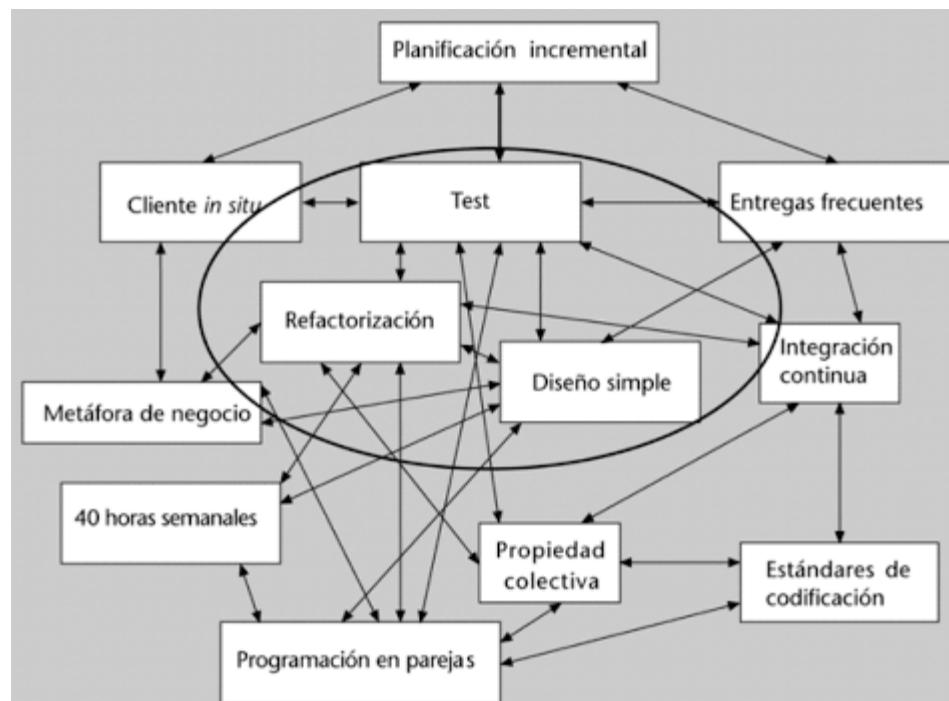
- Es una mejora sobre la programación en parejas, que amplía esta manera de trabajar más allá del ámbito del desarrollo, hacia el coaching, la gestión o la definición de historias de usuario (storytelling).

Usabilidad iterativa:

- Busca ser una mejora sobre la involucración del cliente, de manera que se evita plantear un diseño complejo desde el comienzo, evolucionando el mismo desde el punto de vista de la usabilidad a medida que se obtiene retroalimentación de los usuarios a través de las diferentes iteraciones.

8. Valoración crítica de XP

Los beneficios que persigue XP a través de los principios, valores y prácticas concretas que propone resultan indiscutibles.



8. Valoración crítica de XP

Recomendaciones frente a la refactorización:

- **Adoptar la propiedad del código compartida**, que facilita que tengamos acceso a todo el código y podamos valorar las implicaciones de los cambios.
- **Seguir estándares de codificación**, de manera que no sea necesario mejorar el estilo antes de la refactorización y que los cambios producidos sean más comprensibles por otros miembros del equipo.
- **Programar en parejas**, de manera que los cambios introducidos sean simultáneamente supervisados por otro compañero.
- **Mantener un diseño simple**, que facilite y al mismo tiempo promueva la refactorización.
- **Disponer de un sistema de integración continua y test automatizados**, de manera que cualquier posible problema derivado de los cambios sea detectado de manera prematura.

8. Valoración crítica de XP

- *Buscar que el programador se encuentre descansado y con la claridad de ideas suficiente para no cometer errores en la refactorización.*

Criticas a XP

XP no está exenta de críticas, se argumenta que mientras que muchas de las prácticas son beneficiosas, algunas otras plantean problemas y el resto simplemente están sobrevaloradas (Rosenberg y Stephens, 2003).

La interdependencia entre las prácticas constituye su gran fortaleza y su principal debilidad, puesto que parece exigir que el correcto funcionamiento de XP implica la adopción de todas las prácticas, cosa que no siempre sucede en todas las organizaciones (bien sea por falta de recursos, o por simple desconocimiento).

8. Valoración crítica de XP

Criticas a XP

Las principales críticas que afronta XP son las siguientes (Pressman, 2010; Stephens, 2003):

- **Variabilidad de requisitos y alcance.** El hecho de involucrar al cliente dentro del equipo de desarrollo puede hacer que los cambios de requisitos se soliciten de manera demasiado informal, y que resulte difícil mantener el alcance del proyecto acotado. Los defensores de XP responden que este problema puede suceder con cualquier otra metodología, dependiendo de las características del proyecto.
- **Expresión informal de los requisitos.** Relacionado con el problema anterior (y en cierto modo en su mismo origen) está el hecho de que la única expresión de las necesidades del cliente son las historias de usuario y las pruebas de aceptación. Los críticos echan en falta una especificación formal, mientras que los defensores argumentan que la naturaleza cambiante de los requisitos en ciertos proyectos vuelve inoperable esta aproximación.

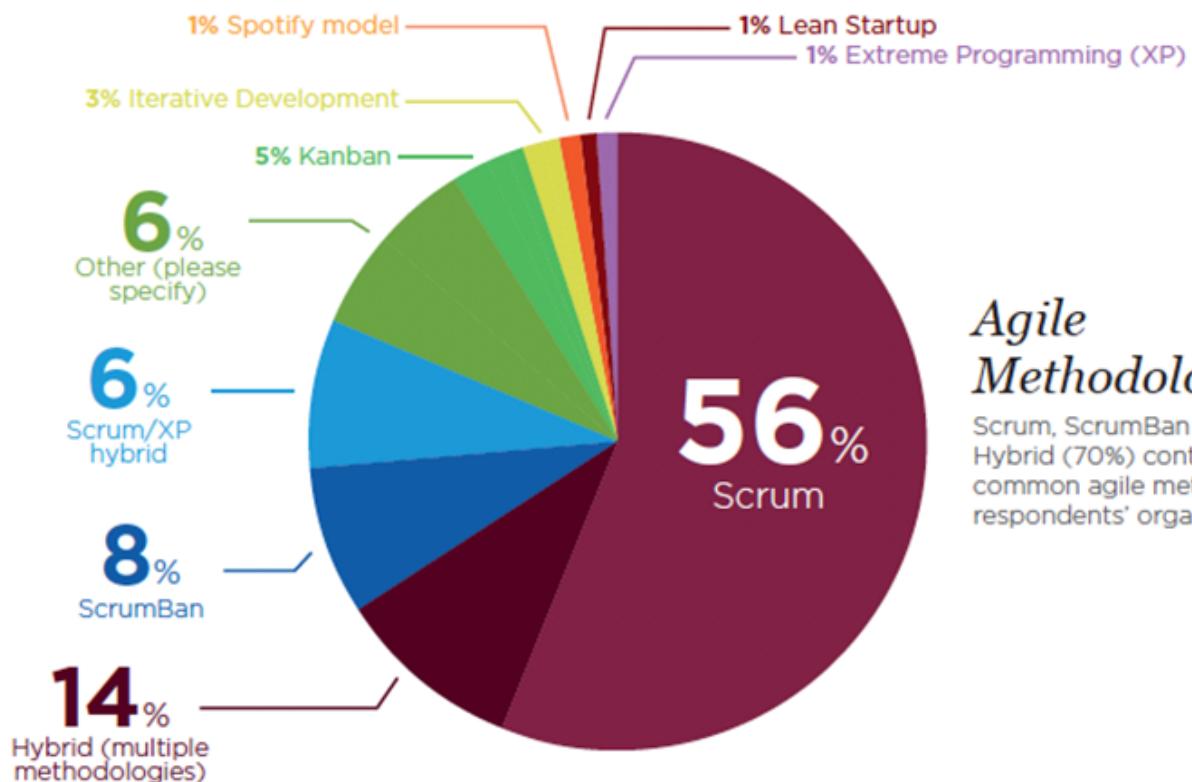
8. Valoración crítica de XP

Criticas a XP

- **Conflict en las necesidades del cliente.** Existen proyectos con varios clientes, con necesidades y objetivos que pueden estar en conflicto. XP no plantea una figura clara que sea capaz de moderar estas posibles discrepancias, puesto que el diálogo se establece directamente entre clientes y equipo de desarrollo, y sucede que esta labor de mediación está más allá de la autoridad y competencias de los desarrolladores.
- **Carenzia de un diseño formal de partida.** Los críticos defienden que el desarrollo de sistemas complejos pasa por un análisis detallado previo que genere un diseño robusto fácil de mantener. Los defensores indican que el proceso incremental de desarrollo, con pequeñas entregas, está orientado a mantener la sencillez de un diseño que se va refinando y analizando en las sucesivas iteraciones, a través de las actividades de diseño parcial y refactorización.

8. Valoración crítica de XP

Criticas a XP



Agile Methodologies Used

Scrum, ScrumBan and Scrum/XP Hybrid (70%) continue to be the most common agile methodologies used by respondents' organizations.

8. Valoración crítica de XP

Criticas a XP

Son muchos quienes consideran que XP es una ***metodología que ha caído en desuso.***

Entre las razones que podemos encontrar para este declive están las siguientes, todas ellas relacionadas con su dificultad de adopción (Partogi, 2016):

Está demasiado centrada en tareas de ingeniería.

- Mientras que el concepto tradicional de metodología de desarrollo se enfoca hacia las tareas de gestión, en XP se da una importancia fundamental a las prácticas de ingeniería del software.
- Muchos jefes de proyecto dejaron abandonadas las tareas de programación en los inicios de sus carreras (en algún caso nunca las realizaron), de manera que XP y su nivel de exigencia desde el punto de vista técnico les asusta.

8. Valoración crítica de XP

Criticas a XP

- Este tipo de manager tradicional tiene difícil encaje en cualesquiera de los roles que propone XP.
- Este tipo de perfiles se encuentra más cómodo con metodologías como Scrum, en la forma de Scrum Master (aunque este rol no es en absoluto equivalente al de jefe de proyecto).

Requiere de un nivel de inversión elevado.

- Esto se debe fundamentalmente al nivel de capacidad técnica esperado del equipo de desarrollo que eleva los costes de personal.
- Los programadores deben estar familiarizados con las prácticas de XP, ser capaces de analizar y comprender diferentes tipos de código, diseñar el sistema, definir y ejecutar pruebas...

8. Valoración crítica de XP

Criticas a XP

- Esto no es habitual encontrarlo en programadores experimentados, y mucho menos entre los recién graduados.
- En caso de no encontrar el perfil, aumentan los costes de formación.
- Los costes de infraestructura también sin importantes dada la necesidad de realizar pruebas automatizadas e integración continua.
- Todo ello sin mencionar los difícil y costoso que es encontrar un buen coach de XP, por el perfil de “artesano de software” que se le demanda (Varios autores, 2009).

8. Valoración crítica de XP

Criticas a XP

Resulta irracional desde el punto de vista de negocio.

- Las personas que toman decisiones en las organizaciones están más próximas al mundo del negocio que al mundo de la ingeniería.
- Para ellos, prácticas como las pruebas unitarias (que implican trabajo adicional y más líneas de código), o la programación en parejas (que puede parecer un despilfarro de recursos) resultan simplemente irracionales.

Es demasiado complejo.

- En comparación con otras metodologías, como Scrum, XP asigna demasiadas obligaciones para el programador.
- Además, hace demasiado énfasis en las entregas continuas de nuevas versiones, cada semana a ser posible, lo cual puede obligar a limitar a una única historia el incremento generado.

8. Valoración crítica de XP

Criticas a XP

Lo cierto es que dada la complejidad y nivel técnico de exigencia que impone XP, muchas organizaciones prefieren comenzar con metodologías ágiles más sencillas y flexibles, como Scrum, e ir introduciendo, progresivamente, aquellas prácticas de ingeniería de XP que resulten más interesantes en función del proyecto.

9. Referencias

- Beck, K. (1999). Embracing Change with Extreme Programming. *Computer*, 32(10), 70-77.
- Beck, K. (2000). Extreme Programming Explained. Embrace Change. Boston: Addison-Wesleyl.
- Fernández, J. (2012). Introducción a las metodologías ágiles: otras formas de analizar y desarrollar [Material didáctico]. Barcelona: Universitat Oberta de Catalunya. Recuperado de <http://hdl.handle.net/10609/63466>
- Fowler, M. (1999). Extract Function. Recuperado de <https://www.refactoring.com/catalog/extractMethod.html>
- Fowler, M. (2018). Página web de Martin Fowler. Recuperado de <https://martinfowler.com>
- GNOME Project. (2018). Guías de programación de GNOME [Web]. Recuperado de <https://developer.gnome.org/programming-guidelines/stable/>

9. Referencias

- ▶ Industrial Logic. (2004). Test-Driven Management [Web]. Recuperado de <http://industrialxp.org/testDrivenManagement.html>
- ▶ Kerievsky, J. (2005). Industrial XP: Making XP Work in Large Organizations. Cutter Consortium. Agile Project Management, 6(2). Recuperado de <https://goo.gl/X5MMTg>
- ▶ King, P., Naughton, P., DeMoney, M., Kanerva, J., Walrath, K. & Hommel, S. (1997). Java Code Conventions. California: Sun Microsystems, Inc. Recuperado de <http://www.oracle.com/technetwork/java/codeconventions-150003.pdf>
- ▶ Mozilla. (2007, agosto 14). Mozilla Coding Style Guide [Web]. Recuperado de <https://web.archive.org/web/20070814042731/http://www.mozilla.org/hacking/mozilla-style-guide.html>
- ▶ Partogi, J. (2016, junio 11). 5 reasons why eXtreme Programming isn't popular [Web]. Recuperado de <https://medium.com/agility-path/5-reasons-why-extreme-programming-isnt-popular-83790418b901>

9. Referencias

- ▶ Pressman, R. S. (2010). Ingeniería del software: un enfoque práctico (7^a ed.). México: McGraw-Hill Educación.
- ▶ Rosenberg, D. & Stephens, M. (2003). Extreme Programming Refactored. The Case Against XP. New York: Apress.
- ▶ Stephens, M. (2003). The case against Extreme Programming [Web]. Recuperado de http://www.softwareality.com/lifecycle/xp/case_against_xp.jsp
- ▶ van Rossum, G. & Warsaw, B. (2001). PEP 7: Style Guide for C Code [Web]. Recuperado de <https://www.python.org/dev/peps/pep-0007/>
- ▶ van Rossum, G., Warsaw, B. & Coghlan, N. (2013). PEP 8: Style Guide for Python Code [Web]. Recuperado de <https://www.python.org/dev/peps/pep-0008/>
- ▶ Varios autores (2009). Manifesto for Software Craftsmanship [Web]. Recuperado de <http://manifesto.softwarecraftsmanship.org/>

9. Referencias

- VersionOne. (2018). 12th Annual State of Agile Report. VersionOne y CollabNet. Recuperado de <https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report>
- Wells, D. (1999). User Stories. Recuperado de <http://www.extremeprogramming.org/rules/userstories.html>
- Wells, D. (2013). Extreme Programming: A Gentle Introduction. Recuperado de <http://www.extremeprogramming.org/>