Clase 2

Programación extrema

(Parte 1)

Nacimiento

La programación extrema o e***X***treme ***P***rogramming es una metodología ágil de desarrollo de software formulada por **Kent Beck**, autor del primer libro sobre la materia: "Extreme Programming Explained: Embrace Change (1999)".

Diferencia

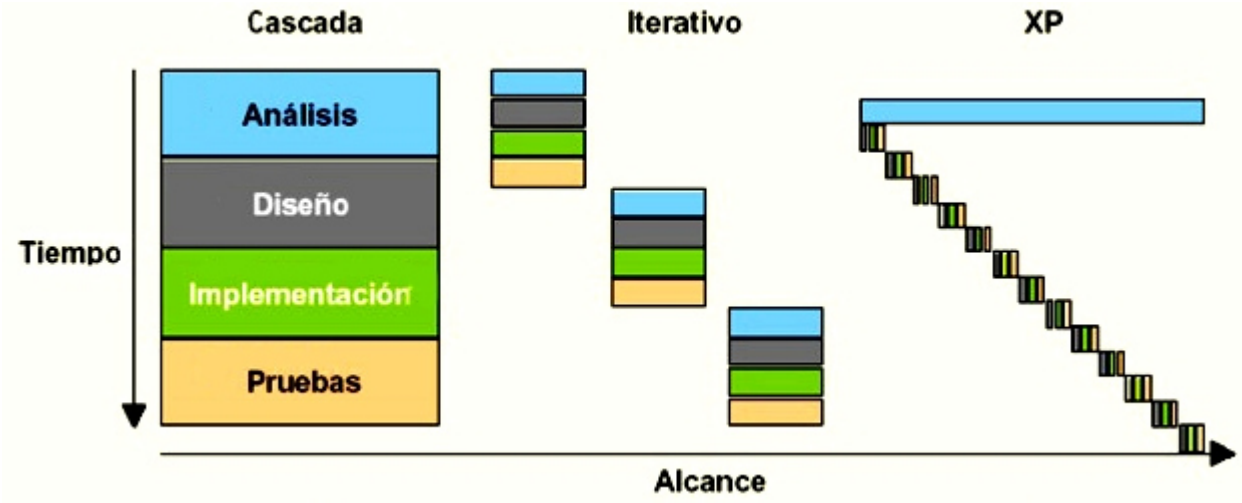
Al igual que otros métodos ágiles, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la **adaptabilidad** que en la **previsibilidad**.
XP considera que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos.

Adaptación

Los defensores de XP sostienen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios.

2. Fases

Un proyecto con XP lleva normalmente 10 a 15 ciclos o iteraciones.

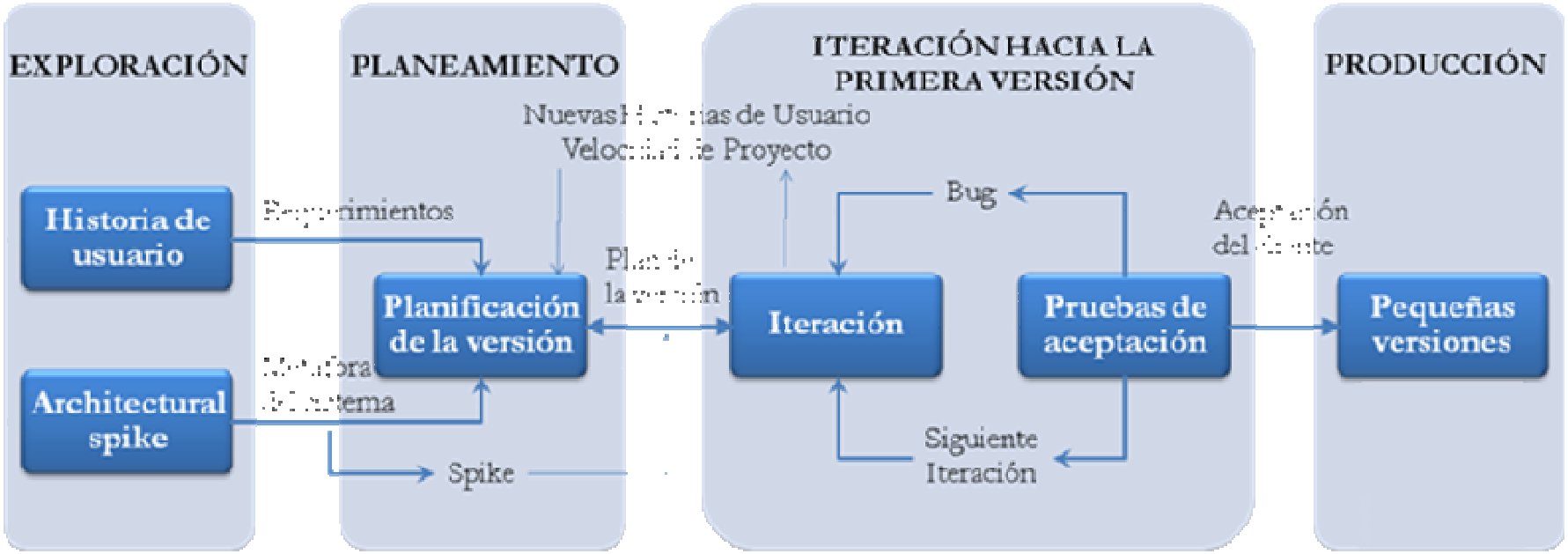


2. Fases

Se definen varias fases, que en general son cuatro:

- **Exploración**
- **Planificación**
- **Iteraciones**
- **Producción**

2. Fases


EXPLORACIÓN
PLANEAMIENTO
ITERACIÓN HACIA LA PRIMERA VERSIÓN
PRODUCCIÓN
Historia de usuario
Architectural spike
Planificación de la versión
Iteración
Pruebas de aceptación
Pequeñas versiones
Bug
Spike
Siguiente Iteración

2. Fases

2.1. Fase de exploración

Es la fase en la que se define el **alcance general del proyecto**. El cliente define lo que necesita mediante la redacción de sencillas "**historias de usuario**". Los programadores estiman los tiempos de desarrollo en base a esta información.

2. Fases

2.1. Fase de exploración

Debe quedar claro que las estimaciones realizadas en esta fase son primarias, ya que están basadas en datos de muy alto nivel, y podrían variar cuando se analicen en más 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.

2. Fases

2.2. Fase de planificación

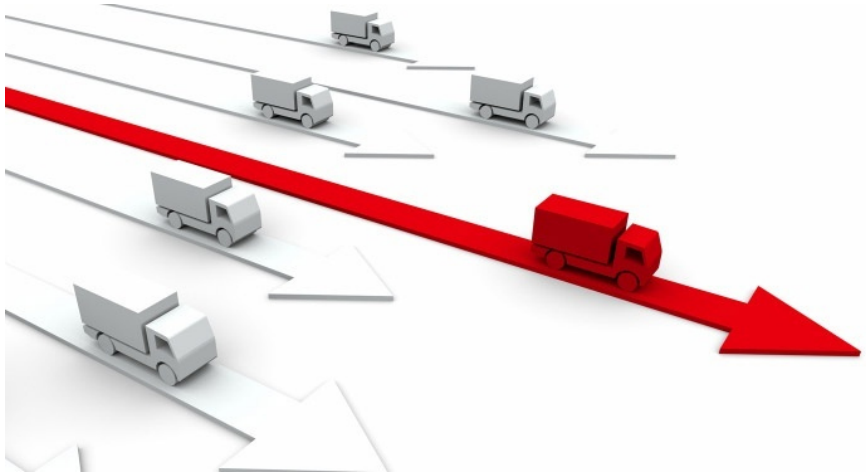
Es una fase corta, en la cual el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario y, asociadas a éstas, las entregas.

2. Fases

2.2. Fase de planificación

Consiste típicamente en una o varias reuniones grupales de planificación.

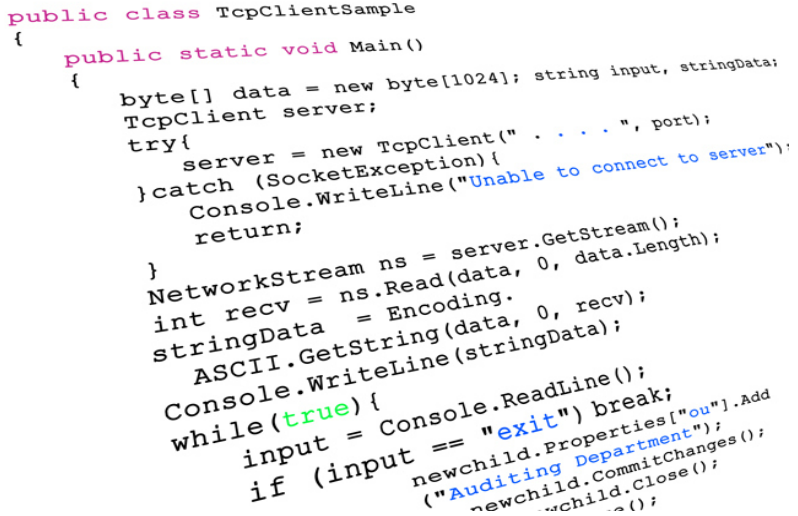
El resultado de esta fase es un **Plan de Entregas**.



2. Fases

2.3. Fase de iteraciones

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.



2. Fases

2.3. Fase de iteraciones

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.

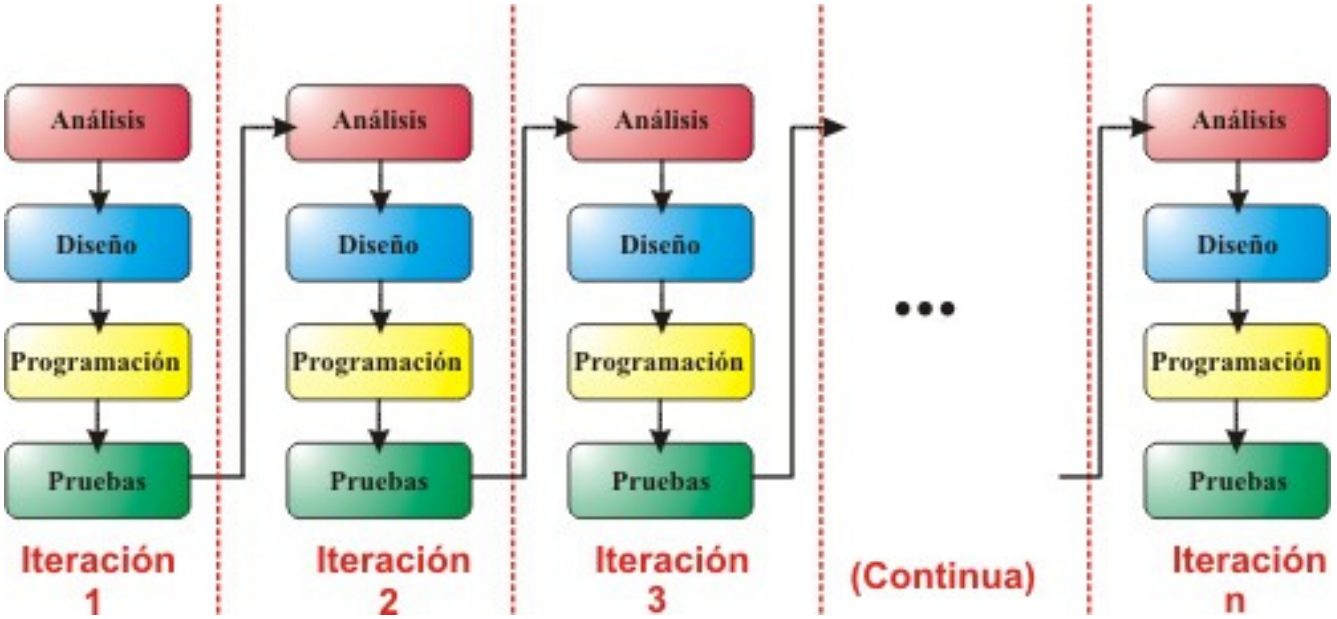
2. Fases

2.3. Fase de iteraciones

Las iteraciones son también utilizadas para medir el progreso del proyecto: una iteración terminada sin errores es una medida clara de avance.

2. Fases

2.3. Fase de iteraciones



2. Fases

2.4. 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.

3. Conceptos

La metodología XP plantea la planificación como un **diálogo continuo** entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes.

3. Conceptos

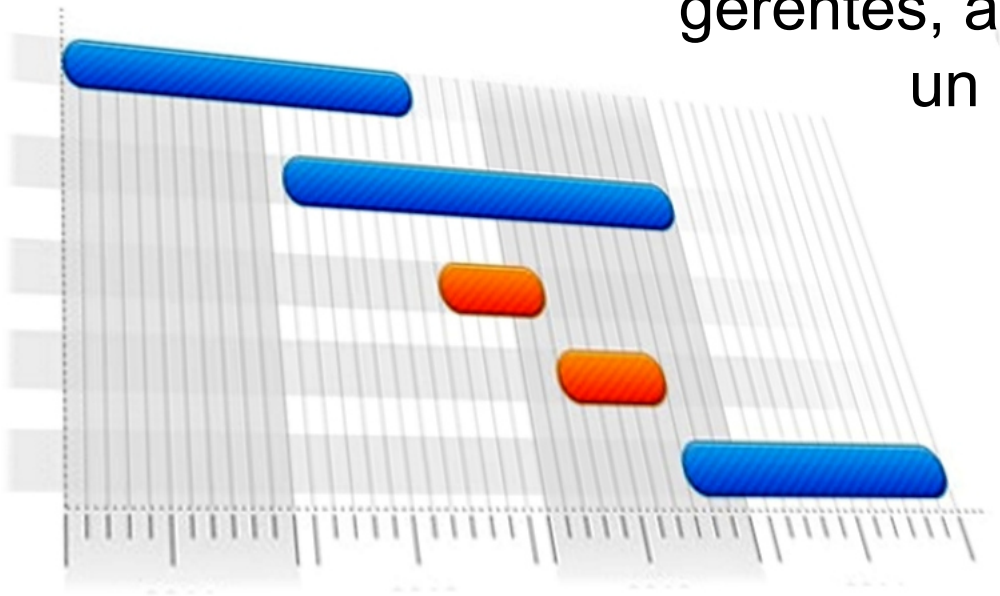
Historia de Usuario	
Número: **1**	Nombre: **Ingreso o supresión de Roles**
Usuario: **Administrador**	
Modificación de Historia Número:	Iteración Asignada:
Prioridad en Negocio: **Baja** (Alta / Media / Baja)	Puntos Estimados:
Riesgo en Desarrollo: (Alto / Medio / Bajo)	Puntos Reales:
Descripción: **En la administración del sistema tendrá la opción de administrar usuarios, al ingresar a esta opción se desplegará un listado de los usuarios, los usuarios van a tener la opción de asignar roles, el administrador hace clic sobre esta opción relacionada con el usuario y el sistema le despliega el listado de roles disponibles para que el administrador seleccione los adecuados para ese usuario. Una vez el usuario administrador del sistema de la opción de guardar, el sistema pide confirmación y luego procederá a almacenar los cambios.**	
Observaciones:	

3. Conceptos

Una vez obtenidas las historias de usuario, los programadores evalúan rápidamente el tiempo de desarrollo de cada una.
Si alguna de ellas tiene "**riesgos**" de no establecer con certeza la complejidad del desarrollo, se realizan pequeños programas de prueba o "**spikes**", para reducir estos riesgos.

3. Conceptos

Completadas estas estimaciones, se organiza una reunión de planificación, con los diversos actores del proyecto: clientes, desarrolladores, gerentes, a los efectos de establecer un cronograma de entregas o "**release plan**" en los que todos estén de acuerdo.



3. Conceptos

Una vez acordado este cronograma, comienza una fase de iteraciones, en dónde en cada una de ellas se desarrollan, prueban e instalan unas pocas historias de usuario.

3. Conceptos

Los planes en XP se diferencian de las metodologías tradicionales en tres aspectos:

1) **Simplicidad del plan**. No se espera que un plan requiera de un “gurú” con complicados sistemas de gerenciamiento de proyectos.

3. Conceptos

Los planes en XP se diferencian de las metodologías tradicionales en tres aspectos:

2) Los planes son realizados por las **mismas personas** que realizarán el trabajo.

3. Conceptos

Los planes en XP se diferencian de las metodologías tradicionales en tres aspectos:

3) Los planes no son predicciones del futuro, sino simplemente **la mejor estimación** de cómo saldrán las cosas.
Los planes necesitan ser cambiados si las circunstancias lo requieren. De otra manera, el plan y la realidad no coinciden, y en estos casos el plan es totalmente inútil.

3. Conceptos

3.1. Historias de usuario

Las “historias de usuario” sustituyen a los documentos de especificación funcional y a los “casos de uso”. Estas “historias” son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar.

3. Conceptos

3.1. Historias de usuario

La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido.
Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo.

3. Conceptos

CONCEPTO	CASOS DE USO	HISTORIAS DE USUARIO
Objetivo	Modelar la interacción entre un 'actor' y el Sistema	Redactar una descripción breve de una funcionalidad tal y como la percibe el usuario
Estructura	Texto detallado donde se sigue una plantilla predefinida a completar con conceptos técnicos (objetivo, resumen, actor, evento disparador, extensiones, etc.)	Corta y consistente en una o dos frases escritas en el lenguaje del usuario.
Planificación	No se utilizan para planificar	Se utilizan para planificar
Agilidad	Requieren tiempo para análisis y la redacción de plantillas predefinidas	Se pueden escribir en pocos minutos
Comprensión	Suelen ser de difícil comprensión, incluso para personal técnico	Fáciles de leer y comprender
Mantenimiento	Suelen pertenecer a documentos con cientos de páginas. Difíciles de mantener.	Muy fáciles de mantener
Comunicación	Modelo textual asociado con diagramas: todo tiene que estar escrito	Basada en la comunicación verbal y orientada a la colaboración y discusión para clarificar detalles

3. Conceptos

3.1. Historias de usuario

CONCEPTO	CASOS DE USO	HISTORIAS DE USUARIO
Soporte	Escritos en documentos con el objetivo de que estos sean archivados como documentos de referencia.	Escritas en tarjetas (teóricas o reales) con el objetivo de que sean usadas directamente
Duración	Puede ser implementada en varias iteraciones	Debe ser implementada y probada en una única iteración
Autores	Definidos por 'intérpretes' (analistas, consultores, etc.)	Posibilidad de ser definidas por usuarios y clientes
Pruebas	La definición de pruebas se redacta en documentación separada	Contienen 'Pruebas de Aceptación' en el reverso de la tarjeta
Contexto	Proporcionan una visión más general del Sistema y la integración en él.	Proporciona una visión menos obvia, por eso las pruebas y el *feedback* de los usuarios son tan importante en las metodologías agiles.
Metodología	Asociado con RUP	Asociado con Programación Extrema (aunque pueden usarse en RUP)

3. Conceptos

3.1. Historias de usuario

Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios.

3. Conceptos

3.1. Historias de usuario

Las historias de usuario deben poder ser programadas en un tiempo entre una y tres semanas.
Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias.
Si es menos de una semana, se debe combinar con otra historia.

3. Conceptos

3.2. Plan de entregas

El cronograma de entregas establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas.
Este cronograma será el resultado de una reunión entre todos los actores del proyecto: cliente, desarrolladores, gerentes, etc.

3. Conceptos

3.2. Plan de entregas

XP denomina a esta reunión "**Juego de planeamiento**" o *"Planning game"*, pero puede denominarse de la manera que sea más apropiada al tipo de empresa y cliente, por ejemplo, "reunión de planeamiento", *"planning meeting"* o *"planning workshop"*.

3. Conceptos

3.2. Plan de entregas

Típicamente el cliente ordenará y agrupará según sus prioridades las historias de usuario. El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores.

3. Conceptos

3.2. Plan de entregas

Después de algunas iteraciones es recomendable realizar nuevamente una reunión con los actores del proyecto, para evaluar otra vez el plan de entregas y ajustarlo si es necesario.

3. Conceptos

3.3. Plan de iteraciones

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido.

3. Conceptos

3.3. Plan de iteraciones

Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. Asimismo, para cada historia de usuario se establecen las pruebas de aceptación.

3. Conceptos

3.3. Plan de iteraciones

Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores.

3. Conceptos

3.3. Plan de iteraciones

Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir.

3. Conceptos

3.4. Reuniones diarias de seguimiento

El objetivo de tener reuniones diarias es mantener la comunicación entre el equipo, y compartir problemas y soluciones. En la mayoría de estas reuniones, gran parte de los participantes simplemente escuchan, sin tener mucho que aportar.

3. Conceptos

3.4. Reuniones diarias de seguimiento

Para no quitar tiempo innecesario del equipo, se sugiere realizar estas reuniones en círculo y de pie.

4. Diseño

La metodología XP hace especial énfasis en los diseños simples y claros.
Los conceptos más importantes de diseño en esta metodología son:

- **Simplicidad**
- **Metáforas**
- **Solución *"spike"***
- **Refactorización**

4. Diseño

4.1. Simplicidad

Un diseño simple se implementa más rápidamente que uno complejo.
Por eso XP propone implementar el diseño más simple posible que funcione.
Se sugiere nunca adelantar la implementación de funcionalidades que no correspondan a la iteración en la que se esté trabajando.

4. Diseño

4.1. Simplicidad

Características fundamentales del código:

- Testeable
- Legible
- Comprensible
- Explicable

4. Diseño

4.2. Metáforas

Una “metáfora” es algo que todos entienden, sin necesidad de mayores explicaciones.

4. Diseño

4.2. Metáforas

La metodología XP sugiere utilizar este concepto como una manera sencilla de explicar el propósito del proyecto, y guiar la estructura y arquitectura del mismo.
Por ejemplo, puede ser una guía para la nomenclatura de los métodos y las clases utilizadas en el diseño del código.

4. Diseño

4.2. Metáforas

Tener nombres claros, que no requieran de mayores explicaciones, redunda en un ahorro de tiempo.

4. Diseño

4.2. Metáforas

Es muy importante que el cliente y el grupo de desarrolladores estén de acuerdo y compartan esta “metáfora”, para que puedan dialogar en un “mismo idioma”.
Una buena metáfora debe ser fácil de comprender para el cliente y a su vez debe tener suficiente contenido como para que sirva de guía a la arquitectura del proyecto.

4. Diseño

4.3. Solución "spike"

Una solución *"spike"* solamente aborda el problema en concreto y se aísla de otro tipo de preocupaciones.

4. Diseño

4.4. Refactorización

La recodificación consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad, a los efectos de hacerlo más simple, conciso y/o entendible.

4. Diseño

4.4. Refactorización

Muchas veces, al terminar de escribir el código de un programa, pensamos que si lo comenzáramos de nuevo, lo hubiéramos hecho de manera diferente, con más claridad y eficiencia.
Las metodologías de XP sugieren recodificar cada vez que sea necesario.

4. Diseño

4.4. Refactorización

Si bien puede parecer una pérdida de tiempo innecesaria en el corto plazo, los resultados de esta práctica tienen sus frutos en las siguientes iteraciones, cuando sea necesario ampliar o cambiar la funcionalidad.

5. Variables

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

- **Costo**
- **Tiempo**
- **Calidad**
- **Alcance**

5. Variables

Tres de ellas podrán ser fijadas arbitrariamente por actores externos al grupo de desarrolladores: usuarios y jefes de proyecto.
El valor de la variable restante lo establecerá el equipo de desarrollo, en función de los valores de las otras tres.
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.