



T4: Extreme Programming

Objetivos y Variables de Control

- **objetivos** → satisfacción cliente
Potenciar al max el trabajo en grupo
- **4 Variables de control**
 - **coste**
 - **calidad**
 - **Ámbito** (cliente, ~~resto~~ en user-stories)
 - Si se fijan 3 **ámbito**, fija la **4^a**
 - Si se intenta fijar las 4 → sobre **CALIDAD**
 - Relación entre las variables: Altamente **NO lineal**
 - XP recomienda → control **ámbito** (cliente>resto)

4 Valores que fomenta XP

Comunicación
 • directa y continua
 → se integra en el equipo

Sencillez
 • Desarrollar solo lo necesario

Realimentación
 • incremental en portes Pequeños
 • entregas **frecuentes** **continuas**
 • permite **reajustar** **agenda** **plumificación**

Valentía
 • saber tomar decisiones difíciles
 • Reparar código cuando se detecta
 • Mejorar código tras realimentación
 • cada vez problema → tratar rápidamente un

Respeto + en 2^a edición
 • **respeto** y **trabajo**
 • nunca entregar cambio y rompa construcción del **sistema**
 • **respetar experiencia** y viceversa
 • equipo desarrollo → derecho a querer **autoridad** sobre su **TI**

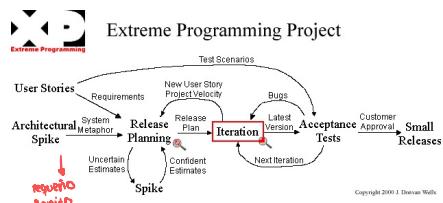
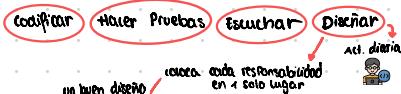
LOS 5 Principios fundamentales XP

- 1) Realimentación Rápida
- 2) Asumir la Sencillez
- 3) Cambio Incremental
- 4) Aceptar Cambio
- 5) Trabajar con calidad

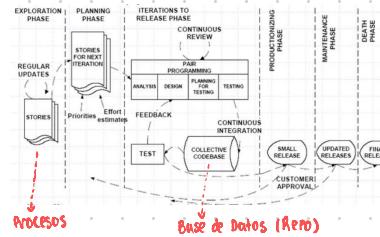
otros principios

- + Enseñar a Aprender
- + Jugar a Ganar ("ganar el juego")
- + Comunicación **Abierta**, **honesta**.
- + trabajar a favor de los instintos de la **ft** (no en vs)
- + Aceptar responsabilidad
- + Ir ligero de Equipo
- + Mediciones honestas
- + Adopción Particular
- + Experimentos Genuinos
- (**17 riesgo decisión** con **truellos**)
- + huir de medidas tipo "100 líneas de código"

4 actividades Básicas



Ciclo de Vida del Proceso XP



"Analiza un poco, diseña un poco, Programa un poco, prueba un poco"

- Empezar por corregir más importante
- Mantener servicios
- Comunicarse y prestar atención a todo el mundo

Las 12 Prácticas de XP

- 1) El juego de la Plumificación
- 2) Entregas Pequeñas
- 3) Metáfora
- 4) Diseño Simple
- 5) Pruebas
- 6) Recodificación
- 7) Programación en Parijas
- 8) Propiedad Colectiva

- 9) Integración continua
- 10) Semana 40h
- 11) Cliente in situ
- 12) Estándares de Codificación

① El juego de la Plumificación

Decisiones de negocio (CLIENTE)

- ALIANZA: cuando una persona sea trabajada en producción?
- PRIORIDAD: priorizar la implementación de las historias
- UNIFICACIÓN: ¿Qué es necesario para que el negocio sea mejor que tener el otro?
- FECHA DE ENTREGA: matrícula. Alumnos **una** **planificación**

Decisiones técnicas (CLIENTE)

- ESTIMACIONES: implementar, revisar...
- CONSECUENCIAS
- PROCESO
- PLANIF. DETALLADA: solo como punto

"Stand - up - Meeting"

- todo el equipo < problemas soluciones >
- todo durante largas reuniones separadas

② Entregas Pequeñas

- poner rápidamente en producción idea buena
 - ↳ nuevos entregas / ciclos cortos
- cada entrega < propia medida
 - ↳ requisitos más básicos + riesgo (**desarrollar con cliente**)
- No realizar requisitos a medias

③ Metáfora

- cada proyecto → guiado por una metáfora global
 - diseñar nombres **variables** usar vocabulario que el cliente entienda
- Proporcionar integridad conceptual

Métodos como "gestionar"
 Objetos como "Cosa"

T4 : Extreme Programming

(Las 12 Prácticas de XP)

④ Diseño Simple

- "La cosa más sencilla que puede hacer es..."
 - uso tarjetas CRC
 - [Class Responsibility Collaborator]
 - DISEÑO ALGÚN CORRECTO DE UN SW
 - Supera todos las pruebas
 - no tiene duplicado
 - Pone en manifiesto vulnerabilidades
 - Alguna n° clases y métodos

⑤ Pruebas.

- Una caract. sin prueba automática asociada, simplemente ~~es~~
 - Los pruebas unitarias ~~son~~ ANTES q el código
 - ↳ pruebas automatizadas ~~garante~~ **TDD**
 - Permiten el desarrollo ~~frente~~ **rápido**, **seguro**.
 - ↳ Equipo de desarrollo → pruebas unitarias
 - ↳ Cliente → pruebas FUTILES
 - Programación, q. ~~seguir~~ del código

⑥ Recodificación

- Recodificación = mejoría del código
 - Antes de añadir un cambio j
 ¿ A alguna forma modificar el código
 existente o hacer más fácil añadir cambio?
 - Despues de añadir un cambio j
 ¿ Se puede simplificar el programa?
 ↳ eliminar complejidad innecesaria
 ↳ eliminar duplicados código
 - Si lo se retro dif'ia cuando el siv. lo requiere
 El cliente no quiso nada

⑦ ↗ Programación en Parejas

- Todo el código se ~~es~~ en Parejos
 - g → Piensa la táctica implementación
 - g → Piensa estrategia
 - é buena aproximación?
 - é va a fallar alguna prueba?
 - é se puede traer algo mejor?
 - Debe realizarse dinámicamente
 - Cambio de Parejos
 - codgo + calidad
 - conocimiento

tracking!

→ fnt R30	→ mejor (?)
→ fnt 1/30	mentres todo el
→ fnt 0h	siglo viene

⑤ Propiedad Colectiva

- Cuál quiera puede modificar en cualquier elemento
 - ↳ evita el Cualos \Rightarrow de botella.
 - ↳ si: $g \in B$ Problema \rightarrow $B \subseteq E$ \rightarrow despl.
 - todo equipo desarrolla  Algunas responsabilidades
corre algo sobre todos pres
cuando muy bien en Pro
 - Cualos se evita mediante pruebas automáticas

⑨ Integración continua

- Código se integra + prueba varias veces / día
 - ↳ Requiere proceso siempre automático
 - ↳ Necesita BIST. Control de versiones
 - 3.1 dedicado para la integración
 - Integración termina a través 100% Pruebas
 - ↳ NO integran código q no pasa las pruebas
 - ↳ SIEMPRE se desprueba de código q pasa



⑩ Semana de 40 horas

- descansados : en productividad

11 Cliente in-situ

- cliente real: el usuario el sistema. Matrículas

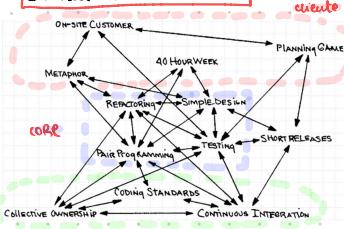
↓
secretaría Alumnos

 - ↳ Responder preguntas
 - ↳ Resolver dudas
 - ↳ Establecer prioridades
 - ↳ Executar tareas
 - ↳ Dar respuestas inmediatas
 - ↳ Enviar q el desarrollo y separar de lo q necesita

⑫ Estándares de codificación

- Cambios de Pareja, Recodif.
 - Código → = estilos, homogéneos + legible
 - Estándar → TDD

Interacción 12 Prácticas



A diagram illustrating the relationship between XP values and practices. At the top, a large red arrow points downwards from the text "2º edición Extreme Programming". Below this, two columns of concepts are aligned. The left column contains four concepts: "3º valor" (with a red arrow pointing to it), "Prácticas", "Primarios", and "corolario". The right column contains three concepts: "error plena/ más sensible" (with a red arrow pointing to it).

PRINCIPIOS DE LA 2º EDICIÓN

- Humanidad
 - Economía
 - Beneficio Mutuo
 - Pequeños Pasos
 - Autosimilaridad
 - Mejora
 - Diversidad
 - Responsabilidad
 - Reflexión
 - Flujo
 - Oportunidad
 - Aceptación
 - Redundancia
 - Fracaso
 - Calidad

prácticas Primarias 2º Edición

- 1) Sentarse Juntos
 - 2) Equipo completo (nuevo talento)
 - 3) Espacio de ^{100%} informativo
 - 4) Trabajo Activo (Energized Work)
 - 5) Programación en Parejas
 - 6) Históricos (stories)
 - 7) Ciclo Semanal
 - Plante., escribir rot., codificar
 - 8) Ciclo ^{10 minutos} temas, DPs, reparaciones, ...
 - 9) Hojuela (^{5 min}) saber q tiene haber implementado y revisar sobre estrategia
 - 10) Construcción inmediata (^{10-min base})
 - 11) Integración continua
 - 12) Programación dirigida, Por Puebas
 - 13) Trabajo incremental

Prácticas corolario 2º Edición

- ② Cliente Real Invio Liderado
 - ③ Despliegue incremental
 - ④ Continuidad del equipo
 - ⑤ Recalificación del equipo
 - (si hay rebajo de trabajo no trabajarlos a todos sino desearcargar a uno)
 - ⑥ Análisis Causa Primaria
 - Preguntarse 5 veces, "Pd. no se detectó" un defecto hasta la causa primaria
 - ⑦ Código compartido
 - ⑧ Código y pruebas
 - ⑨ Base de código única
 - ⑩ Despliegue diario
 - ⑪ Contrato de ámbito negocial fijar G, coste, calidad NO ámbito
 - ⑫ Pago Por Uso
 - "el E es la reeliminación fundamental"

T4: Extreme Programming

Comparación con las 12 Prácticas

- 1) El juego de la planificación → más serena (ciclo vida < 6 meses)
- 2) Entregas pequeñas → más explícitas Desarrollo < diseño
- 3) Metáfora, no documente
- 4) Diseño Simple → Diseño incremental + Base de código única (con menor modulos)
- 5) Pruebas → Programación dirigida por pruebas
- 6) Recodificación → (implícita) Diseño incremental
- 7) Program en Parejas → =
- 8) Propiedad colectiva → Código compacto
- 9) Integración continua → =
- 10) Semana de 40 horas → Trabajo activo + Holgura en ciertas tareas
- 11) Cliente in-situ → Soñarse juntos + Equipo formado → cliente real involucrado
- 12) Estándares de tipificación → se deduce < código compatible program. en parejas



Algunas estrategias de XP

Instalaciones físicas, planificación, desarrollo, diseño, pruebas

Instalaciones físicas

Planta abierta sin grandes luces
Periferia: mini cubículos trabajo individual (pizarras)
Centro: PC's rodeados (mesa)



Desarrollo

Planificación de iteraciones

- Tarjetas de tareas (ver

Task	Service for Task	Start Date	End Date
Card No:	IED Plan	1	1/1/00
Start	IED Plan	1	1/1/00
End	Activity	1	
Description:	Create new service for assigning track number and partner name to a track		
Update NewTrack to create track id		1	
Create new service to get unique track id		1	
Update IED ID for AddTrack		1	
Update AddTrack and ChangeTrack related		1	
Run all test		1	
Update Release & deploy properties		1	
Release code and companion assets and metrics		1	

Exploración / tareas en 2 etapas
combinar/descombinar

Compromiso → ideal
factor larga (1 o 2 programando)
Balanceado (larga total/programador)

Implementar

Ajuste → revisar
Recuperación
Verificar

Programación en Parejas

Planificación

- ① Objetivos
 - unir equipo a tarea común
 - decidir alcance
 - prioridades
 - estimar coste
 - planificación
 - Dar confianza → Sist. debe hacerse
 - Buena base para implementación

- ② Principios
 - Asumir simplicidad (detalle solo prox horizonte)
 - Responsabilidad aceptada
 - Responsable tarea → estimar
 - Ignorar dependencias entre tareas
 - Priorizar prioridades

Inicio

Interacción con el cliente → requisitos
historias de usuarios
(en h)

User-story

Story Title:	Generate a Unique Track Number	Priority:	H
Creation Date:	10/25/00	Risk:	M
Card No.:	11.7	IED#:	3

Description: There is a unique number assigned to each music track or media within a partner space so that music tracks or media are unique across Partners, and the Track Source ID is unique across all Partners.

• When a track is added its number is validated for uniqueness and to fit within 28 bits.

- The TrackSourceID is now:
 - 2 bits for Version
 - 12 bits for Partner Number
 - 28 bits for Track Number
- The Activity log parsing is revised for this new schema.

Completion Criteria: Verify that a unique number is generated when a new track is added, and it is 28 bits long. The TrackSourceID is unique and complies with the definition, and is properly parsed and written to the Activity Log.

Validación

Planificación de entregas

- Muchos y frecuentes
- En iteraciones (1 → 2 o 3 semanas)
- Grandes tareas

Diseño XP

- Simples y en pruebas
- Revisar y mejorar continuamente

Mejor diseño

- Pasa todos pruebas
- No código duplicado
- Expresar ideas que se quieren comunicar
- Menor nº de errores

Role de los diagramas

- Visión ligera
- Trabajar con los clientes

Visión común → vocab. compartido

- Ayudar a crear el BSR
- PQA solo como herramienta (Metacora)

El juego de la Planificación

Cliente → Desarrolladores

Objetivo: Maximizar valor software

Estrategia: Bueno, Bonito, Barato

Actividades

- Exploración: cliente B historia →
- Compromiso: cliente C historia →
- Ajustar: cliente D historia →

Iteración Recuperación Nueva list Resimulación

T4 : Extreme Programming

(Algunas estrategias de XP)

Pruebas

Reglas para el desarrollo de Pruebas:

- ① Se programan a la vez q el código
- ② Cada prueba es independiente
- ③ Pruebas deben ser Automáticas
- ④ Funciona / No funciona
- ⑤ Solo probar → romper integridad test
- ⑥ Deben querer 100% correctitud

Pruebas de desarrollo

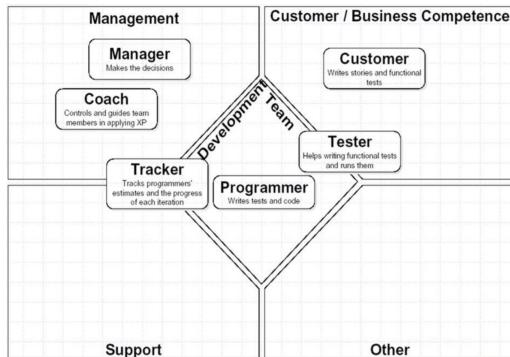
- ① Pruebas unitarias antes del código si:
 - ↳ Método no claro
 - ↳ Implementación parece complicada
 - ↳ Circuito: Alarma
- ② Aparece problema → ③ prueba **✓** eliminación
- ④ Recodificar → (antes hacer pruebas)

Pruebas cliente → funcionales

(otros tipos de prueba...)

- ① Pruebas de regresión
 - ↳ Comparar funcionalidades en otra versión
- ② Pruebas de stress
 - ↳ Sist. Por situaciones de carga posible
- ③ Prueba de Atoms (Memory testing)
 - ↳ Ingresar datos sin sentido
 - ↳ Verificar la resistencia del sist. a datos erróneos

Roles en XP



Programador:

- Poco Pensable decisiones técnicas
- Responsable construir sistema
- NO distinción entre **Análisis** (**Diseñadores**) y **Implementación** (**Programadores**)
- Programan, diseñan, testean

Pero todos tienen q ser de todo (=)

Jefe de Proyecto: organiza y guia reuniones
asegura condiciones adecuadas

Cliente: parte del equipo
determina **QUÉ** y **CUANDO** construir
Establece pruebas funcionales

Tester: Ayuda a pruebas finales
Asegura pruebas finales **✓**

Postrevisor: **Metric Man**
Observa sin molestar
Conserva datos históricos

Entrenador: responsable proceso
Figura 2º Plano a medida que el equipo madura

Consultor: Apoyo y consultoría en temas específicos

Como soluciona XP los problemas

- Retrasos y desviaciones en la planificación **<** versiones cortas entre entregas pensadas
- Costes de Mantenimiento muy elevado → Pruebas continuas
- Alta tasa de defectos → Pruebas continuas
- Requisitos Mal comprendidos → cliente dentro equipo
- Cambios de negocio no reflejados en el SW → versiones cortas
- Falsa rigidez de características → se usan tareas Prioritarias
- Rotación de Personal → Anima el contacto e integración

¿Cómo adoptar XP?

Adoptar 1 Práctica a la vez

- 1) Eleger 1º problema
- 2) Resolvérlo estilo XP
- 3) Eleger otro problema (1)