4.3 Metodologías ágiles de desarrollo de SW

XP

Indice

XP

"El énfasis está en **adaptar el proyecto** – lo cual es bastante sencillo – en lugar de dar una predicción exacta de qué se necesitará y cuánto tiempo hará falta – lo cual es muy difícil"

Ron Jeffries, co-fundador XP

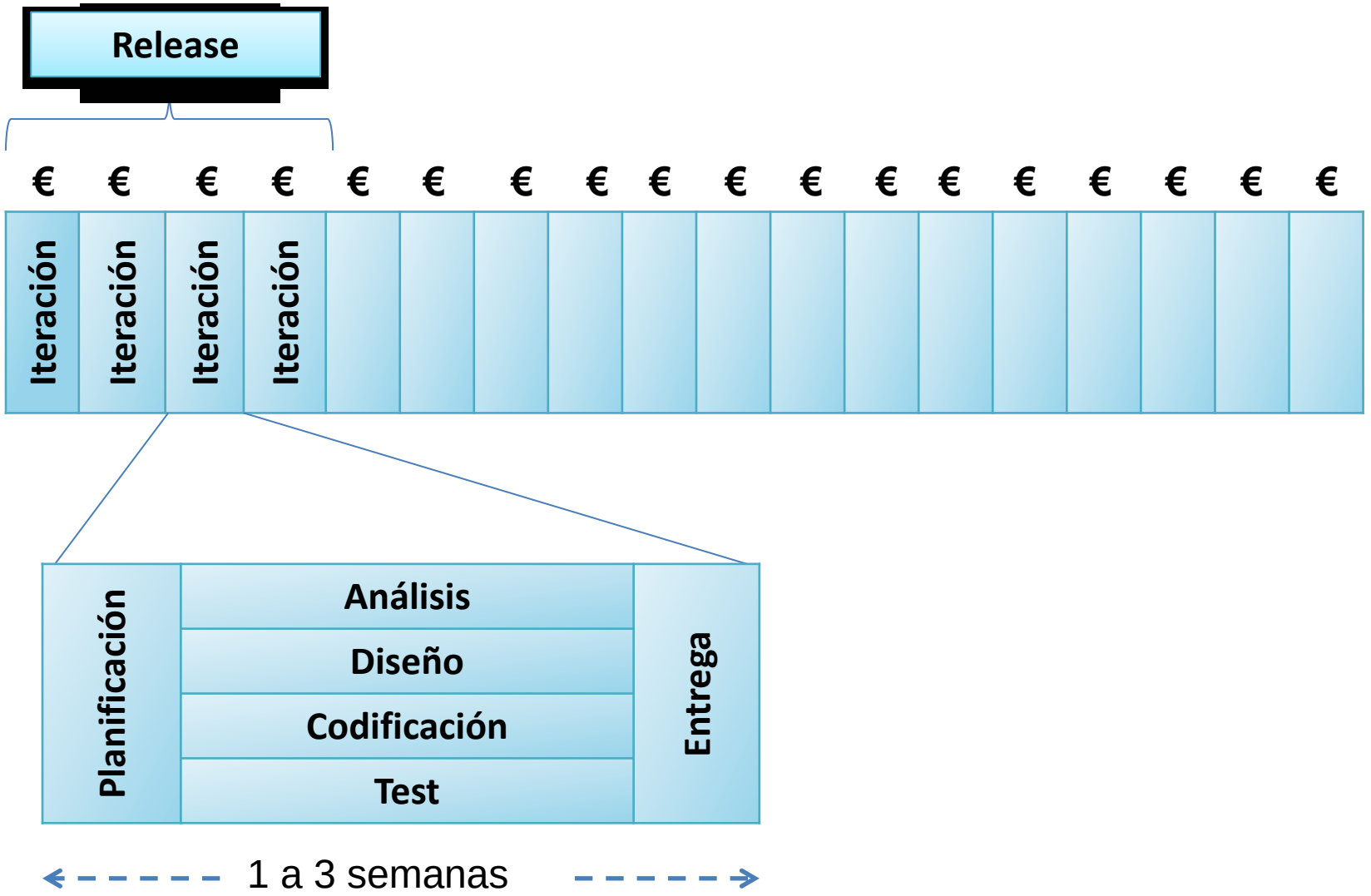
¿Qué es XP?

Metodología desarrollada por:

Kent Beck

- eXtreme Programming (XP) es una metodología centrada en potenciar las relaciones interpersonales como clave para el éxito
- Debe existir una realimentación continua entre el cliente y el equipo de desarrollo
- Esta metodología es adecuada para proyectos con requisitos imprecisos y muy cambiantes y donde existe un alto riesgo técnico

Ciclo de vida en XP



Roles en XP

Programador

- Escribe las pruebas unitarias y produce el código del sistema
- Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo

Cliente

- Escribe las historias de usuario y las pruebas funcionales para validar su implementación
- Asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio

Roles en XP

Encargado de pruebas (*Tester*)

- Ayuda al cliente a escribir las pruebas funcionales
- Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas

Encargado de seguimiento (*Tracker*)

- Proporciona realimentación al equipo
- Debe verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones

Roles en XP

Entrenador (*Coach*)	• Es responsable del proceso global • Guía a los miembros del equipo para seguir el proceso correctamente
Consultor	• Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto • Guía al equipo para resolver un problema específico
Gestor (*Big boss*)	• Dueño del equipo • Es el vínculo entre clientes y programadores • Su labor esencial es de coordinación

Ciclo de desarrollo en XP

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

Ciclo de desarrollo en XP

- En todas las iteraciones tanto el cliente como el programador aprenden
- El ciclo de vida ideal de XP consiste en 6 fases:
  1. Exploración
  2. Planificación de la entrega (release)
  3. Iteraciones
  4. Producción
  5. Mantenimiento
  6. Muerte del proyecto

Fase 1: Exploración

- Los clientes plantean a grandes rasgos las historias de usuario de interés para la primera entrega del producto
- El equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto
- Se prueba la tecnología y se construye un prototipo
- Esta fase transcurre de pocas semanas a pocos meses

Historias de usuario

- Técnica utilizada en XP para especificar los requisitos del software
- Son tarjetas de papel donde el cliente describe brevemente las características del sistema, sean requisitos funcionales o no funcionales
- Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en un par de semanas como máximo

Historias de usuario

- Cada historia de usuario tiene una prioridad asociada definida por el cliente
- El tiempo de cada historia de usuario es determinado por los desarrolladores
- Cada historia debe tener pruebas de validación

Historias de usuario

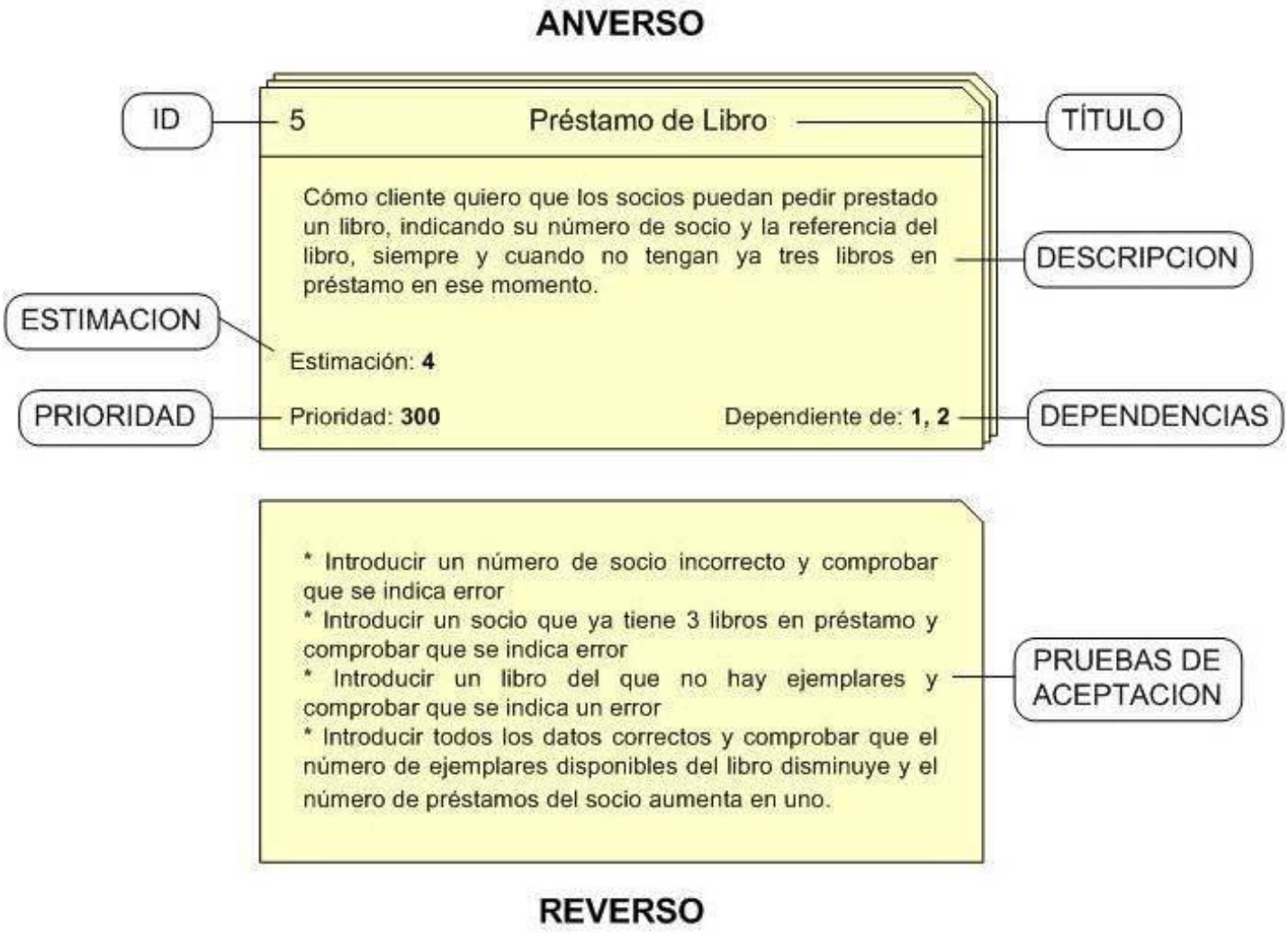
- Las historias de usuario deben responder a 3 preguntas:
  - Quién se beneficia
  - Qué se quiere
  - Cuál es el beneficio
- Se recomienda redactar las historias de usuario con el siguiente formato:
  - Como (rol) quiero (algo) para poder (beneficio)

Historias de usuario

Características:

- Independientes unas de otras
- Negociables
- Valoradas por los clientes o usuarios
- Estimables
- Pequeñas
- Verificables

Historias de usuario



Fase 2: Planificación de la entrega

- El cliente establece la prioridad de cada historia de usuario
- Los programadores realizan una estimación de esfuerzo para cada una de ellas
- Se decide sobre el contenido de la primera entrega
- Una entrega debería proporcionarse entre 2 y 6 meses
- Esta fase dura unos pocos días

Fase 2: Planificación de entrega

- Las estimaciones de esfuerzo de los programadores se realizan utilizando como medida el punto
- Un punto, equivale a una semana ideal de programación
- Las historias generalmente valen de 1 a 3 puntos
- El equipo de desarrollo mantiene un registro de la velocidad de desarrollo establecida en puntos por iteración
- La velocidad del proyecto se usa para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo llevará implementar un conjunto de historias

Fase 2: Planificación de la entrega

- La planificación se puede realizar basándose en el tiempo o en el alcance
- Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar
- Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

Fase 3: Iteraciones

- Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado
- El Plan de entrega está compuesto por iteraciones de no más de 3 semanas
- En la primera iteración se puede establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto
- Al final de la última iteración el sistema estaría listo para entrar en producción

Fase 3: Iteraciones

- Elementos a tener en cuenta en el plan de la iteración:
  - Historias de usuario no abordadas
  - Velocidad del proyecto
  - Pruebas de aceptación no superadas en la iteración anterior
  - Tareas no terminadas en la iteración anterior
- Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas asignadas a un programador como responsable, pero llevadas a cabo por parejas de programadores

Fase 4: Producción

- Esta fase requiere de pruebas adicionales y revisiones de rendimiento antes de trasladar el sistema al entorno del cliente
- Se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase
- Es posible que se rebaje el tiempo de cada iteración, de 3 a 1 semana
- Las ideas propuestas y las sugerencias se documentan para su posterior implementación (por ejemplo, durante la fase de mantenimiento)

Fase 5: Mantenimiento

- Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones
- Tareas de soporte para el cliente
- La velocidad de desarrollo puede bajar después de la puesta del sistema en producción
- La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura

Fase 6: Muerte del proyecto

- El cliente no tiene más historias para incluir en el sistema
- Se deben satisfacer las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema
- Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura
- La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo

Prácticas en XP

El juego de planificación

- Cerrar iteraciones entre el cliente y el programador
- Los programadores estiman el esfuerzo para la implementación de las historias de los clientes
- El cliente decide acerca del alcance y el tiempo de las entregas

Entregas pequeñas y frecuentes

- Un sistema simple es producido rápidamente (al menos una vez cada 2 o 3 meses)
- Pueden liberarse nuevas versiones diariamente pero al menos se debe liberar una cada mes

Prácticas en XP

Metáforas del sistema

- El sistema es definido por una metáfora o conjunto de metáforas entre el cliente y los programadores
- Esta “historia compartida” guía todo el desarrollo describiendo cómo trabaja el sistema

Diseño simple

- Enfasis en diseñar la solución más simple posible que sea implementada en el momento
- Complejidad innecesaria o código extra son eliminados inmediatamente

Prácticas en XP

Pruebas

- El desarrollo del sistema es conducido por pruebas continuas (TDD: Test Driven Development)
- Los clientes ayudan a escribir pruebas funcionales antes de comenzar a escribir el código
- El propósito real del código no es cumplir un requerimiento sino pasar las pruebas

Refactorización

- Reestructurar el sistema eliminando duplicidad, mejorando la comunicación, simplificando y añadiendo flexibilidad
- Mejora la estructura interna del código sin alterar su comportamiento

Prácticas en XP

Programación por pares

- Dos personas escriben el código en el mismo ordenador
- El que no está escribiendo piensa desde un punto de vista más estratégico y realiza lo que podría llamarse revisión del código en tiempo real
- Se pueden turnar varias veces al día

Propiedad colectiva del código

- Cualquier persona puede cambiar cualquier parte del código en cualquier momento
- Siempre debe escribirse antes la prueba correspondiente

Prácticas en XP

Integración continua

- Una pieza nueva de código se integra en el sistema tan pronto como esté lista
- El sistema se integra y construye muchas veces al día
- Todas las pruebas deben ejecutarse antes y después de la integración

Ritmo sostenible

- Se trabaja semanas de 40 horas
- El desarrollo de software se considera un ejercicio creativo por lo que se debe estar descansado para hacerlo eficientemente

Prácticas en XP

Todo el equipo trabajando junto

- Es preferible que el equipo de desarrollo este en un espacio de trabajo abierto
- Los programadores por pares deberían ubicarse en el centro del espacio
- El cliente debería estar presente y disponible para el equipo (Cliente in situ)

Prácticas en XP

Estándares de programación

- Reglas de código de notación, indentación y nomenclatura seguidas por todos los programadores
- El código debe ser auto documentado (llamado “código revelador de intenciones”). Si el código es tan confuso que necesita comentarios, se debe reescribir o refactorizar

Principios básicos

Simplicidad

- Es la base de la programación extrema
- Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento
- Un diseño complejo del código junto a sucesivas modificaciones por parte de diferentes desarrolladores hacen que la complejidad aumente exponencialmente
- Para mantener la simplicidad es necesaria la refactorización del código
- El código debe estar auto documentado (elegir adecuadamente nombres de variables, métodos y clases)

Principios básicos

Comunicación

- Para los programadores el código comunica mejor cuanto más simple sea
- Debe comentarse sólo aquello que no va a variar, por ejemplo, el objetivo de una clase o la funcionalidad de un método
- Las pruebas unitarias son otra forma de comunicación
- Los programadores se comunican constantemente gracias a la programación por parejas
- La comunicación con el cliente es fluida, ya que, el cliente forma parte del equipo de desarrollo
- El cliente decide qué características tienen prioridad y siempre debe estar disponible para solucionar dudas

Principios básicos

Retroalimentación

- Al estar el cliente integrado en el proyecto, su opinión sobre el estado del proyecto se conoce en tiempo real
- Al realizarse ciclos muy cortos tras los que se muestran resultados, se minimiza el tener que rehacer partes que no cumplen con los requisitos y ayuda a los programadores a centrarse en lo que es más importante
- Ejecutar las pruebas unitarias frecuentemente permite descubrir fallos debidos a cambios recientes en el código

Principios básicos

Coraje o valentía

- Hay que ser valiente para confiar en que la programación por parejas beneficia la calidad del código sin repercutir negativamente en la productividad
- Se requiere de coraje para implementar las características que el cliente quiere ahora sin caer en la tentación de optar por un enfoque más flexible que permite futuras modificaciones

Errores comunes y malas interpretaciones

- Cliente no in situ. Uso de especificaciones escritas para la siguiente iteración
- Aplicar sólo algunas de las prácticas de XP
- Xp es sólo un desarrollo iterativo + documentación mínima + pruebas unitarias
- No es necesario escribir las pruebas antes de comenzar a programar
- El cliente no decide
- No realizar pruebas de aceptación escritas por el cliente

Errores comunes y malas interpretaciones

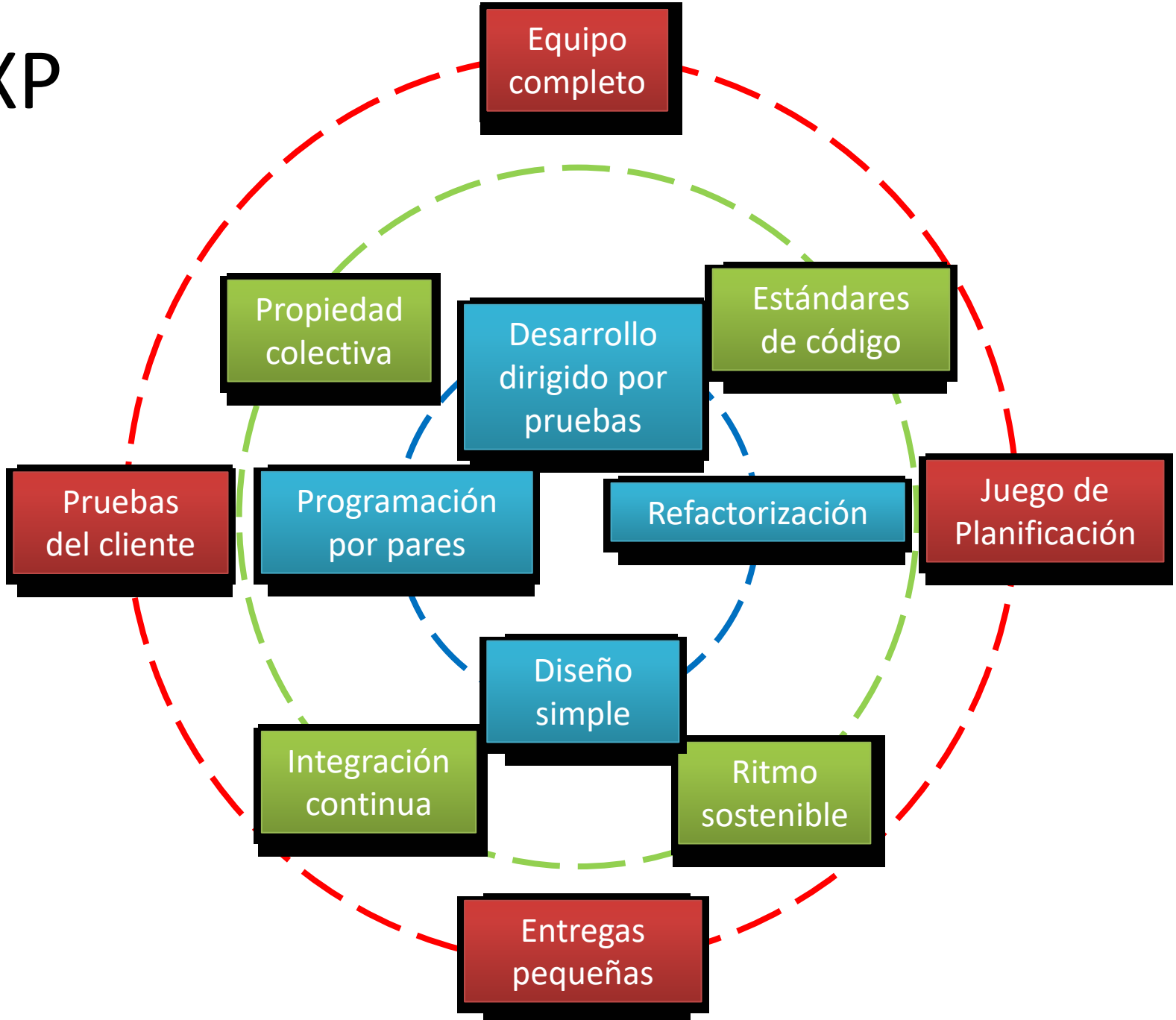
- El cliente no está presente en la revisión de sus pruebas de aceptación
- Sólo puede haber un cliente in situ
- Muchas tarjetas de tareas demasiado detalladas
- Programar con el mismo compañero mucho tiempo
- El cliente o el manager realizan seguimientos continuos

Errores comunes y malas interpretaciones

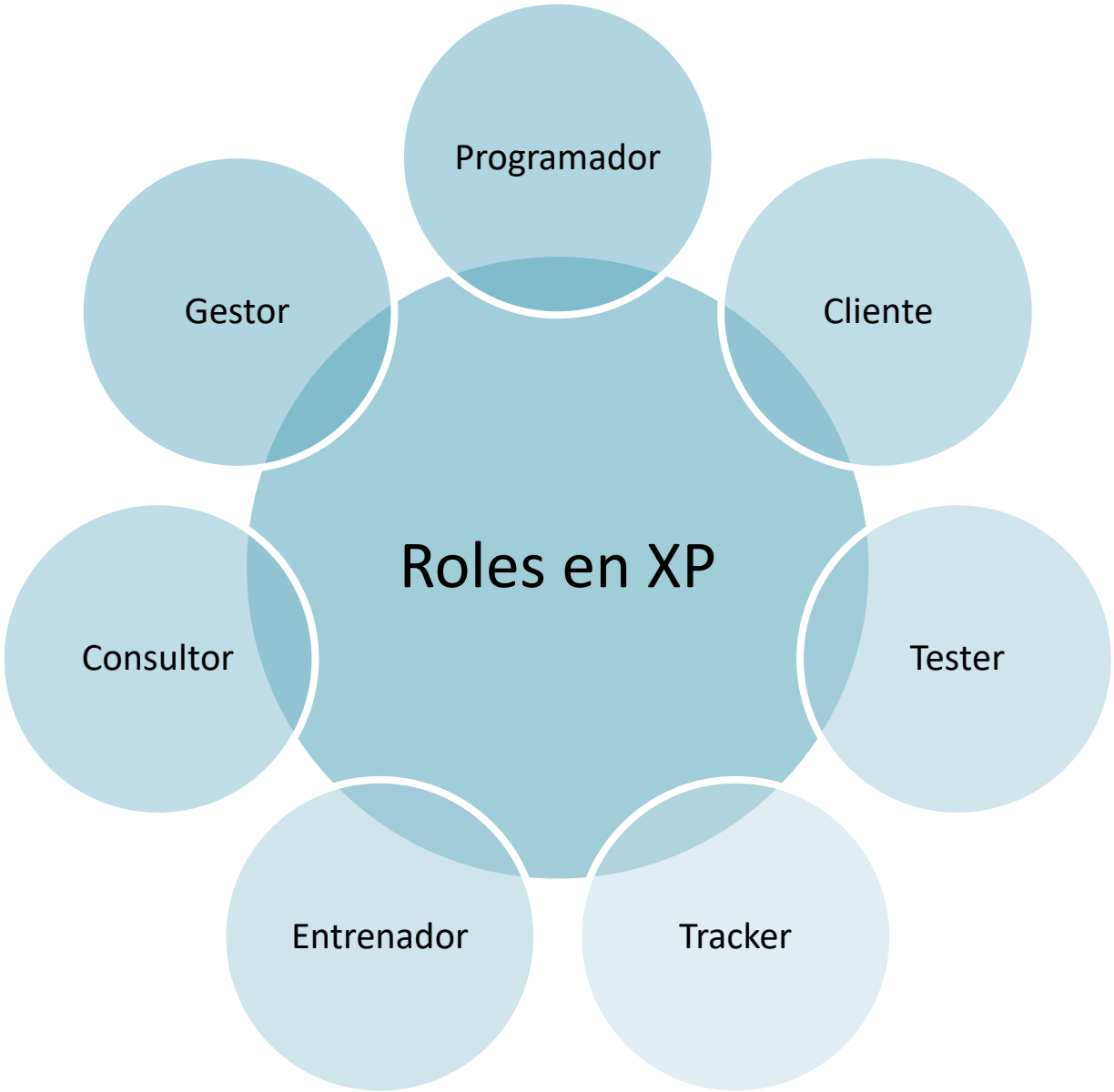
- Hacer diagramas es malo
- Usar programadores sin experiencia en el trabajo por pares
- Iteraciones demasiado largas
- Una iteración no acaba con una versión integrada y probada
- Cada iteración obtiene una versión lista para su producción
- Planificación predictiva

RESUMEN

XP


Equipo completo
Propiedad colectiva
Desarrollo dirigido por pruebas
Estándares de código
Pruebas del cliente
Programación por pares
Refactorización
Juego de Planificación
Diseño simple
Integración continua
Ritmo sostenible
Entregas pequeñas

XP


Programador
Gestor
Cliente
Roles en XP
Consultor
Tester
Entrenador
Tracker

Bibliografía

- Extreme Programming Explained. Kent Beck. Addison Wesley
- Extreme Programming in Practice. James Newkirk & Robert C. Martin. Addison Wesley
- The Art of Agile Development. James Shore & Shane Warden. O'Reilly