# Relación de ejercicios 5

**Javier Gómez Luzón**

**1. El problema básico del desarrollo del software es el riesgo, algunos ejemplos de situaciones de riesgo son:**

- **Retrasos de planificación: llega el día de la entrega, y le decimos al cliente que el software no estará disponible.**

Cuando se percata de que el proyecto se está retrasando, el hecho de trabajar tiempo extra es perjudicial, el trabajo extra desmotiva al grupo y repercute en la calidad del proyecto. Se debería, en medida de lo posible, renegociar el plan de entregas.

- **Proyecto cancelado: después de numerosos retrasos, el proyecto se cancela sin haber entrado nunca en producción.**

Con XP se aconseja que el cliente elija la versión más pequeña y operativa que tenga sentido para el negocio, ahorrando así fallos antes de empezar a producir.

- **El sistema se deteriora: el software se pone satisfactoriamente en producción, pero después de un par de años, los costes de hacer cambios o la tasa de defectos crecen tanto que el sistema debe ser reemplazado.**

XP tiene una serie de pruebas que se ejecutan unas cuantas veces al terminar un cambio en el software para evitar un deterioro del mismo.

- **Tasa de defectos: el software se pone en producción, pero la tasa de defectos es tal alta que no se usa.**

XP realiza un gran número de pruebas. Desde la perspectiva de los programadores y desde la perspectiva de los clientes.

- **Requisitos mal comprendidos: el software se pone en producción, pero no resuelve el requisito planteado inicialmente.**

XP incita al cliente a que forme parte del equipo, asegurando así una buena comunicación entre el cliente y el equipo, consiguiendo que todo se entienda correctamente.

- **Cambios en el negocio: el software se pone en producción, pero el problema de negocio para el que se diseñó la solución inicial fue reemplazado hace seis meses por otro problema más acuciante.**

XP invita a que durante el desarrollo del proyecto, el cliente pueda añadir o quitar ciertas funcionalidades.

- **Falsa riqueza de características: el software tiene un montón de características potencialmente interesantes, todas las cuales fueron divertidas de programar, pero ninguna hace que el cliente gane dinero.**

XP antepone las tareas de alta prioridad.

- **Cambios de personal: después de dos años, todos los buenos programadores del proyecto, comienzan a odiar el programa y se marchan. ¿Cómo trata XP el riesgo que hemos comentado anteriormente?**

XP demanda a los trabajadores compromiso para completar y estimar su trabajo. Consiguiendo así que los trabajadores no se estresen. Promueve también las relaciones entre el equipo para que así nadie se quede aislado.

## 2. Comenta brevemente las diferencias principales entre XP y Scrum

Scrum vs XP
En Scrum las iteraciones son de 2-4 semanas y se conocen como sprint. En XP las iteraciones de entrega son de 1-3 semanas.
En Scrum las tareas realizadas del Sprint Backlog al final de un Sprint, no vuelven a tocarse en el futuro. En XP las tareas terminadas y entregadas al cliente pueden ser modificadas durante el transcurso del proyecto.
En Scrum el product Owner marca el orden de prioridad del Sprint Backlog y el Scrum Team puede seguir ese orden, a no ser que el equipo considere cambiarlo. En XP el equipo de desarrollo sigue este orden estrictamente, además de ayudar a decidir sobre él en el principio.
Scrum es una metodología de desarrollo ágil centrada en cómo administrar un proyecto. El objetivo de XP es la programación o creación del producto final.
En Scrum cada miembro trabaja de forma individual autogestionandose, mientras que en XP se trabaja en parejas.

## 3. Compara los roles de XP con los roles de Scrum.

Tanto Scrum como XP tiene el rol de programador o equipo de desarrollo, que son casi iguales. En Scrum el equipo trabaja individualmente, además de tener más capacidad de decisión; en XP se trabaja en parejas.

En XP y Scrum el cliente, tiene las mismas responsabilidades.

XP tiene al Coach y Scrum a Scrum Master. Ambos tienen responsabilidades parecidas, teniendo el Scrum Master más responsabilidades que el Coach, ya que el Scrum Master debe ayudar también al Product Owner.

XP tiene el Gestor y Scrum tiene el Product Owner. Ambos tienen similares responsabilidades con la diferencia de que el Product Owner debe de hacer de conexión entre el cliente y los programadores, siendo parte del equipo y ayudar al equipo de desarrollo.

## 4. Relacionar los 14 principios de XP2 con las prácticas propuestas en XP.

1. “Sentarse juntos” se relaciona con “Humanidad” y “Beneficio mutuo”.
2. “Integración Continua” se relaciona con “Mejora, redundancia, calidad y fallo”.
3. “Pruebas antes de implementar” se relaciona con “Principios de Fallo y de Redundancia”.
4. “Diseño Incremental” se relaciona con “Principios de Mejora, Calidad y Pasos de bebé”.
5. “Whole Team” (Sensación de equipo) se relaciona con “Principios de humanidad, beneficio mutuo, mejora, reflexión y flujo”.
6. “Informative Workspace” (Espacios de trabajo informativos ) se relaciona con “Principio de Calidad” y con “Beneficio mutuo”.
7. “Quarterly Cycle” (Ciclo trimestral) se relaciona “Principios de redundancia, fallo, reflexión y responsabilidad”.
8. “Slack” (Aflojar) se relaciona con “Principio de reflexión” y con “Principios de fallo y responsabilidad”.
9. “Energized Work” (Trabajo a pleno rendimiento) se relaciona con “Principios de calidad, mejora y responsabilidad”.
10. “Programación en parejas” se relaciona con “Principio de beneficio mutuo” y con “Principios de Calidad, Diversidad, Mejora y Responsabilidad”.
11. “Historias de Usuario” se relaciona con “Principios de Flujo y de Pasos de bebé”.
12. “Weekly Cycle” (Ciclo Semanal) se relaciona con “Principios de redundancia, fallo, reflexión y responsabilidad”.
13. “Ten minute build” se relaciona con “Principios de Economía” y “Principio de flujo”.

## 5. Clasifica las prácticas de XP que hemos comentado en el tema 4 en las siguientes categorías:

- **Planificación y análisis de requisitos.**

-Ciclo semanal.
-Ciclo trimestral.

- **Factores humanos y equipo.**

-Sentarse juntos.
-Programación en parejas.
-Sensación de equipo.
-Aflojar.

- **Diseño.**

-Pruebas antes de implementar.
-Historias de usuario.
-Diseño incremental.
-Construcción en 10 minutos.

- **Codificación del software y entrega.**

-Trabajo a pleno rendimiento.
-Integración contínua.

### 6. ¿Para qué se utiliza WIP (Work in progress) en Kanban?

Los objetivos de Kanban son visualizar y analizar el flujo de trabajo. Ahí es donde entra WIP, en saber que cantidad de trabajo hay en progreso actualmente. Esto ayuda a saber si hay algo que bloquea el desarrollo, el proceso se para en seco y el equipo y la organización se concentran en solucionar el problema. También ayuda a ver la carga del sprint y la velocidad del equipo.

### 7. Comenta brevemente las diferencias entre Kanban y Scrum.

En Kanban no hay roles predefinidos para un equipo, aunque si puede haber un rol de “director del equipo”.
Kanban mide la producción midiendo lo que se tarda en completar una parte completa de un proyecto de principio a fin y en Scrum mide la velocidad a través de Sprint.
En Kanban los productos y procesos se entregan continuamente según sea necesario, en Scrum los entregables se entregan al final de los sprints o periodos de tiempo establecidos en los que se debe completar un conjunto de trabajos.