Tecnicatura Universitaria en Programación
(TUP-2024)

Cátedra de Metodología en Sistemas I
Profesora: Petrelli Nadia

Actividad Nº 3
C9
Grupo 3

Integrantes:

- Díaz Medina Tomás
- Díaz Rossini Juan José
- Gallo Genaro
- Hinojosa Alexis
- Liacoplo Gerónimo Emiliano
- Navarro Victor Leandro

Consigna:

1) Investigar como trabajan otras metodologías ágiles distintas a Scrum y Kanban
2) En grupos asignar roles de scrum
3) Comenzar a crear las cards de forma manual

Desarrollo:

1) OTRAS METODOLOGÍAS ÁGILES DISTINTAS A SCRUM Y KANBAN
    1. XP (Extreme Programming)

    Es una metodología ágil enfocada en la calidad del código y la adaptabilidad. Promueve prácticas como: desarrollo en parejas (pair programming), pruebas automatizadas constantes, integración continua y diseño simple. Es útil en proyectos donde los requerimientos cambian frecuentemente.

XP (Extreme Programming) es una metodología ágil creada por **Kent Beck** a fines de los 90. Su objetivo es mejorar la calidad del software y responder bien a los cambios frecuentes en los requisitos del cliente.
Se basa en la simplicidad, la comunicación constante, el feedback rápido, la valentía y el respeto dentro del equipo.

Valores de XP

1. **Comunicación**
2. **Simplicidad**
3. **Retroalimentación**
4. **Valentía**
5. **Respeto**

Principales prácticas (core practices)

XP tiene 12 prácticas agrupadas en torno a desarrollo, pruebas y colaboración:

Categoría	Práctica	Qué hace
Desarrollo	Diseño simple	Crear lo más simple posible que funcione
	Programación en parejas	Dos personas en una sola máquina, una escribe, otra piensa
	Integración continua	Integrar cambios varias veces al día
	Refactorización	Mejorar el código sin cambiar lo que hace
Pruebas	Desarrollo guiado por pruebas	Escribir tests antes de programar
	Pruebas de aceptación	Validar que el sistema cumple requisitos del cliente
Gestión	Pequeñas entregas frecuentes	Publicar versiones funcionales rápido
	Juego de planificación (Planning Game)	Definir qué se hará en cada iteración
	Ritmo sostenible (no sobretrabajo)	Evitar el burnout del equipo
Colaboración	Cliente siempre disponible	El cliente forma parte activa del equipo
	Propiedad colectiva del código	Cualquiera puede modificar cualquier parte del código
	Estándar de codificación	Todo el código sigue las mismas reglas

Roles en XP

Aunque menos rígidos que en Scrum, sí hay roles definidos:

Rol	Función principal

Cliente (Customer)	Persona del negocio que está disponible y define los requisitos
Desarrolladores	Todo el equipo técnico. Comparten responsabilidad sobre el código
Coach	Ayuda al equipo a entender y aplicar correctamente XP
Tracker	Lleva métricas de progreso, como velocidad y cumplimiento
Tester (opcional)	Asegura que todo se valida y funciona según lo esperado

Ciclo de vida en XP

XP funciona en iteraciones cortas (como sprints) y cada ciclo incluye:

1. Planificación
2. Diseño simple
3. Codificación en parejas
4. Testing automático
5. Refactorización
6. Entrega frecuente

¿Cuándo usar XP?

- Cuando los requisitos cambian mucho
- Cuando hay contacto directo y frecuente con el cliente
- En equipos pequeños y motivados
- Proyectos con alta exigencia de calidad de código

2. Lean Software Development

Derivado del sistema de producción de Toyota, busca maximizar valor eliminando desperdicios. Principios clave: entrega rápida, mejora continua, amplificación del aprendizaje y respeto por las personas. Muy efectivo para proyectos grandes y procesos complejos.

Es una adaptación al desarrollo de software de los principios de producción Lean creados por **Toyota**. En lugar de centrarse sólo en escribir código, **busca maximizar el valor y eliminar el desperdicio** en todo el proceso.

Principio	Explicación breve
Eliminar desperdicio	Evitar código innecesario, documentación inútil o trabajo duplicado
Ampliar el aprendizaje	Validar constantemente el conocimiento mediante prototipos o versiones
Decidir lo más tarde posible	No comprometerse a decisiones importantes sin evidencia
Entregar lo más rápido posible	Evitar esperas. El software útil debe llegar al cliente cuanto antes
Empoderar al equipo	Confianza total al equipo de desarrollo para tomar decisiones
Construir calidad desde el inicio	Prevenir errores en vez de arreglarlos después
Ver todo el sistema	Pensar en todo el proceso, no solo en una parte aislada

Ciclo en Lean:

1. Crear una **hipótesis de valor**
2. Construir lo mínimo necesario (**MVP**)
3. Recibir feedback real
4. Medir impacto
5. Iterar

Roles

No tiene una estructura de roles estricta como Scrum. Se trabaja más horizontalmente, con **líderes de equipo, desarrolladores, testers y clientes integrados**.

¿Cuándo usarlo?

- En equipos maduros y autogestionados
- Proyectos que buscan optimización y entrega rápida
- Cuando se prioriza el valor y feedback continuo sobre el control rígido

3. FDD (Feature-Driven Development)

Se centra en desarrollar software basado en funcionalidades concretas. El proyecto se divide en "features" (funcionalidades), cada una con diseño y planificación independiente. Es útil para proyectos con requisitos claros y estructuras organizativas grandes.

Metodología ágil estructurada que gira en torno a **funcionalidades específicas (features)**. Nació en 1997 con **Jeff De Luca** y **Peter Coad**, ideal para proyectos grandes y organizados.

5 pasos del proceso FDD

1. **Desarrollar un modelo general**
    - Visión de alto nivel del sistema
2. **Construir la lista de funcionalidades (features)**
    - Frases breves que describen acciones concretas ("calcular total", "generar reporte")
3. **Planificar por funcionalidades**
    - Se asignan dueños de cada feature y tiempos de entrega
4. **Diseñar por funcionalidad**
    - Se hace un diseño técnico específico para cada feature
5. **Construir por funcionalidad**

- Se codifica y prueba funcionalidad por funcionalidad

Rol Principal	Función
Project Manager	Supervisa el proyecto general
Chief Architect	Define el modelo general del sistema
Development Manager	Coordina los desarrolladores
Feature Owners	Responsables de cada funcionalidad
Class Owners	Dueños del código en clases específicas
Domain Experts	Clientes o usuarios que definen requisitos funcionales

¿Cuándo usarlo?

- Proyectos **grandes**, con muchos equipos
- Cuando hay **requisitos bien definidos**
- Cuando se necesita **medir el progreso claramente por entregables**

4. Crystal

Es una familia de metodologías que se adapta según el tamaño del equipo y la criticidad del sistema. Hace hincapié en la comunicación y en la interacción humana por encima de los procesos.

Crystal es una familia de metodologías ágiles desarrollada por **Alistair Cockburn**.
Su lema es:
"**Las personas son más importantes que los procesos**".
Cada variante de Crystal se adapta al tamaño del equipo y la criticidad del sistema:

- **Crystal Clear** (1–6 personas)
- **Crystal Yellow** (6–20)
- **Crystal Orange** (20–50)
- Y más...

Principios fundamentales

- **Comunicación frecuente y cara a cara**
- **Reflexión y mejora continua**
- **Entrega frecuente de software funcional**
- **Seguridad (según la criticidad del sistema)**

- Adaptabilidad según el contexto

Enfoque

No establece una única forma de trabajar. Se adapta al proyecto. A diferencia de Scrum, **no impone roles o ciclos fijos**, sino que **cada equipo construye su proceso** según necesidades reales.
Roles mínimos sugeridos:

Rol sugerido	Función
Líder técnico	Guía técnica y decisiones de arquitectura
Desarrolladores	Codifican y prueban
Usuario experto	Da feedback rápido y continuo
Facilitador	Ayuda en comunicación y adaptación del proceso

¿Cuándo usarlo?

- En equipos pequeños o medianos
- Cuando se valora la flexibilidad total
- En proyectos donde **la calidad y comunicación son más importantes que los procesos rígidos**

2) ASIGNACIÓN DE ROLES SCRUM EN GRUPO (ejemplo para 6 integrantes)

- Product Owner: Persona que mejor entienda la necesidad del usuario (quien haya propuesto el proyecto)
- Scrum Master: Persona organizada, con buena comunicación (facilita reuniones y elimina obstáculos)
- Equipo de Desarrollo (Dev Team): los 4 restantes se dividen funciones de programación, diseño, bases de datos, testing, etc.

Product Owner: Navarro Victor Leandro
Scrum Master: Liacoplo Gerónimo Emiliano
Equipo de desarrollo:

- Díaz Medina Tomás
- Díaz Rossini Juan José
- Gallo Genaro
- Hinojosa Alexis

3) CARDS de las Historias de los Usuarios

ID	Título	Rol del Usuario	Deseo (Qué quiere)	Para qué (Objetivo)	Estimación	Prioridad
HU-001	Generador de Personajes	Jugador	Crear personajes completos fácilmente	Ahorrar tiempo siguiendo reglas oficiales	5 puntos	MUST
HU-002	Generador de Objetos	Jugador	Crear objetos siguiendo las reglas	Equipar personajes con ítems personalizados	3 puntos	SHOULD
HU-003	Generador de Hechizos	Jugador	Diseñar hechizos basados en el manual	Usarlos en partidas según su clase	5 puntos	SHOULD
HU-004	Generador de Enemigos	DM	Crear enemigos personalizados	Agilizar la creación de encuentros	5 puntos	COULD
HU-005	Generador de Loot	DM	Crear recompensas automáticas	Generar botines durante la aventura	4 puntos	COULD
HU-006	Generador de Trampas	DM	Diseñar trampas y obstáculos	Agregar dificultad y variedad a la partida	4 puntos	COULD
HU-007	Manejo de Sesiones	DM	Guardar y gestionar la campaña	Tener toda la información segura y compartida	8 puntos	MUST