The New Methodology - IDS1

1. Idea Central y Contexto

Martin Fowler analiza el surgimiento de las **metodologías ágiles** como respuesta a las limitaciones de los enfoques tradicionales (**predictivos** o **orientados a procesos**). Estas metodologías priorizan la **adaptabilidad**, la **colaboración humana** y la **entrega incremental** de software funcional, reconociendo que el desarrollo es un proceso **creativo** e **impredecible**. Su popularidad creció en los 2000 como alternativa a métodos burocráticos como el **Rational Unified Process (RUP)**, buscando equilibrar estructura y flexibilidad en entornos empresariales dinámicos.

---

2. Evolución Histórica: De la Informalidad a lo Ágil

- **Fase 1: "Code and Fix" (Caótica):**
  - Desarrollo sin planificación, común en proyectos pequeños.
  - Problemas: Deuda técnica, pruebas tardías y dificultad para escalar.
- **Fase 2: Metodologías Predictivas (Ej: RUP):**
  - Basadas en **planificación detallada** y **documentación extensa**.
  - Inspiradas en ingeniería civil: separan diseño (UML) y construcción (codificación).
  - **Limitaciones:**
    - Burocráticas y lentas.
    - Asumen requisitos estables, algo poco realista en entornos empresariales.
- **Fase 3: Metodologías Ágiles (Adaptativas):**
  - Surgen para manejar la **incertidumbre** y los **cambios frecuentes**.
  - Ejemplos: Extreme Programming (XP), Scrum, Crystal.

---

3. Diferencias Clave: Predictivo vs. Ágil

Aspecto	Predictivo	Ágil

Planificación	Detallada a largo plazo (ej: diagramas UML).	Iterativa y flexible (ej: sprints de 1-4 semanas).
Documentación	Prioriza documentos formales (requisitos, diseños).	Prioriza código funcional y comunicación directa.
Respuesta al cambio	Resistente (cambios son costosos).	Adaptativa (los cambios son una ventaja).
Roles	Jerárquicos (analistas, desarrolladores, testers separados).	Equipos multidisciplinarios y autoorganizados.
Énfasis	Procesos estandarizados.	Personas y colaboración.

Ejemplo:

En un proyecto predictivo, un cambio en los requisitos durante la codificación requiere revisar documentos de diseño, actualizar planes y retrasar entregas. En un proyecto ágil, el cambio se discute en la próxima iteración y se implementa sin romper el flujo.

---

4. Principios Fundamentales de las Metodologías Ágiles

a) Adaptabilidad sobre Predictibilidad

- **Causas de la impredictibilidad:**
  - Requisitos volátiles (ej: cambios en el mercado).
  - Complejidad técnica emergente.
- **Mecanismos de adaptación:**
  - **Iteraciones cortas:** Entregas frecuentes (ej: sprints en Scrum).
  - **Retroalimentación constante:** Reuniones diarias (daily scrums) y revisiones con el cliente.

b) Orientación a las Personas

- **Crítica al modelo taylorista:**
  - Enfoque tradicional trata a los desarrolladores como "recursos intercambiables".
  - **Consecuencias:** Desmotivación, alta rotación y baja innovación.
- **Valores ágiles:**

- **Autonomía:** Los equipos deciden cómo trabajar (ej: estimaciones técnicas en XP).
- **Responsabilidad profesional:** Desarrolladores como expertos en su campo.

c) Colaboración con el Cliente

- **Roles clave:**
  - **Product Owner (Scrum):** Representa al cliente, prioriza el backlog.
  - **Usuario activo (XP):** Participa en pruebas y definición de criterios de aceptación.
- **Beneficios:**
  - Software alineado con necesidades reales.
  - Menos sorpresas en la entrega final.

---

5. Procesos Autoadaptativos y Mejora Continua

- **Retrospectivas:**
  - Reuniones al final de cada iteración para reflexionar:
    - ¿Qué funcionó bien?
    - ¿Qué podemos mejorar?
  - **Ejemplo:** Un equipo Scrum descubre que las reuniones diarias son demasiado largas y decide limitarlas a 15 minutos.
- **No hay "metodología única":**
  - **Crystal:** Propone metodologías flexibles según el tamaño del equipo y la criticidad del proyecto (ej: Crystal Clear para equipos pequeños).
  - **Context-Driven Testing:** Adapta las pruebas al contexto del proyecto, no a estándares rígidos.

---

6. Metodologías Ágiles Principales

a) Extreme Programming (XP)

- **Pilares:**

    - **Pruebas unitarias (TDD):** Escribir pruebas antes del código.
    - **Integración continua:** Fusionar código diariamente para detectar errores temprano.
    - **Programación en pares:** Dos desarrolladores trabajan juntos en una misma tarea.
- **Valores:** Comunicación, simplicidad, coraje (ej: refactorizar código legacy).

b) Scrum

- **Elementos clave:**
    - **Sprints:** Iteraciones de 2-4 semanas con un objetivo claro.
    - **Roles:** Scrum Master (facilitador), Product Owner, Equipo de desarrollo.
    - **Artefactos:** Backlog (lista de tareas), Tablero Kanban (seguimiento visual).

c) Lean Development

- **Inspiración:** Sistema de producción de Toyota (eliminar desperdicios).
- **Principios:**
    - **Optimizar el flujo:** Reducir tiempos de espera entre etapas.
    - **Decidir lo más tarde posible:** Postergar decisiones irreversibles hasta tener más información.

d) Kanban

- **Enfoque visual:** Tablero con columnas (Por hacer, En progreso, Hecho).
- **Regla:** Limitar el trabajo en progreso (WIP) para evitar sobrecarga.

---

7. Cuándo y Cómo Adoptar Métodos Ágiles

a) Proyectos Adecuados

- **Ideal para:**
    - Requisitos incompletos o cambiantes (ej: startups, innovación).
    - Equipos motivados y colaborativos.
- **No recomendado para:**

- Proyectos con requisitos estables y altamente regulados (ej: software médico).

b) Pasos para Implementar

1. **Seleccionar un proyecto piloto:**
   - Preferiblemente pequeño pero estratégico (ej: funcionalidad crítica de un producto).
2. **Capacitar al equipo y stakeholders:**
   - Talleres sobre valores ágiles y herramientas (ej: Jira para Scrum).
3. **Iniciar con prácticas básicas:**
   - **Ejemplo:** Empezar con reuniones diarias y sprints de 2 semanas.
4. **Iterar y ajustar:**
   - Usar retrospectivas para refinar el proceso.

c) Errores Comunes

- **Imponer prácticas sin consenso:**
  - Si el equipo no está comprometido, las metodologías fracasan.
- **Ignorar la cultura organizacional:**
  - Empresas jerárquicas pueden resistirse a la autonomía ágil.

---

8. Conclusiones y Reflexiones Finales

- **Agilidad no es ausencia de estructura:** Es una estructura flexible, centrada en **valor entregado** y **aprendizaje continuo**.
- **Éxito depende de las personas:** Equipos empoderados, clientes colaborativos y líderes que faciliten (no controlen).
- **No es una solución mágica:** Requiere adaptación constante y voluntad de cambiar.

Frase clave de Fowler:

"El software no es un puente. Es un medio para explorar ideas, y como tal, debe ser maleable."
