Qué es ScrumXP (resumen corto)

ScrumXP no es un nuevo framework mágico: es usar **Scrum** como marco de trabajo para organizar el equipo y la cadencia (sprints, eventos, roles, backlog) y complementar eso con **prácticas técnicas** de eXtreme Programming (XP) — tests automáticos, integración continua, pair programming, refactor, diseño simple, releases frecuentes — para que lo que Scrum planifica termine siendo software de calidad y sostenible. En otras palabras: Scrum da la estructura; XP da la disciplina técnica. (Scrum.org)

Por qué combinarlas (beneficios principales)

- Acelera entrega de valor real y desactiva la deuda técnica (porque XP obliga a tests y refactor).
- Reduce riesgo: releases pequeñas + integración continua hacen que los errores se detecten temprano.
- Mejora predicibilidad: sprint tras sprint con feedback técnico constante.
- Aumenta colaboración: pair programming + revisión continua elevan el conocimiento del equipo.
- Mantiene enfoque en la calidad sin sacrificar velocidad. (Scrum.org)

Prácticas XP que debes conocer (lo esencial)

(implementa la que puedas, pero prioriza **TDD + CI/CD + pair programming**)

- **Test-Driven Development (TDD)** — escribir tests primero, código después.
- **Pair Programming** — dos devs en la misma tarea (driver + navigator).
- **Continuous Integration / Continuous Delivery (CI/CD)** — integra y despliega pequeñas porciones frecuentemente.
- **Refactor** continuo — mejorar el diseño sin cambiar comportamiento.
- **Simple Design** — el diseño más simple que funcione hoy.
- **Collective Code Ownership** — cualquiera puede tocar cualquier parte del código.
- **Small Releases / Frequent Integration** — acortar el tiempo entre idea y producción.
- **Coding Standards** — reglas compartidas para legibilidad y consistencia. (Scrum.org)

Cómo encaja cada cosa con Scrum (práctico)

- **Sprint Planning**: prioriza ítems que tengan en cuenta la capacidad de hacer TDD/automatización.

- **Daily**: además del impedimento, usa el daily para sincronizar prácticas técnicas (p.ej. pairing necesario hoy).
- **Sprint Review**: muestra valor realmente testeado y desplegable.
- **Sprint Retrospective**: usa retro para ajustar prácticas XP (¿más pair? ¿mejor CI?).
- **Definition of Done (DoD)**: incluye criterios XP — por ejemplo "unit tests + integración verde + revisión de código". Esto hace que "done" signifique **entregable y mantenible**. (Scrum.org)

Roadmap de adopción (pasos concretos)

1. **Alineá el equipo en la idea**: Scrum es el marco, XP las prácticas. Hacé una sesión corta para acordar objetivos de calidad.
2. **Definí DoD que incluya prácticas XP** (tests, CI, build verde).
3. **Instalá CI básico** (pipeline que corre tests en cada push).
4. **Empieza con TDD en una feature pequeña** (prueba y aprende).
5. **Promové pair programming en tareas críticas** (no hace falta 100% del tiempo).
6. **Automatizá releases** — un pipeline que pueda desplegar al menos a staging con un click.
7. **Mide, retroalimenta y ajusta**: velocidad, tasa de fallos en producción, tiempo de recuperación. Hazlo iterativo: no intentes imponer TDD + pairing + cambios de arquitectura en una semana; ve sumando prácticas cada sprint. (Scrum.org)

Métricas útiles (no te vuelvas esclavo)

- **Lead time / Cycle time** (idea → producción).
- **% de builds verdes** y **fallos en CI**.
- **Cobertura de tests** (útil como indicador, no como dios absoluto).
- **Defects en producción** y **MTTR** (tiempo medio de reparación).
- **Velocidad del equipo** (pero priorizá calidad, no solo bajar números).

Riesgos y trampas comunes (y cómo evitarlos)

- *Trampa:* "hacemos TDD pero escribimos tests inútiles". → Priorizar tests que aporten valor (integración + unit).
- *Trampa:* usar pair programming todo el tiempo sin rotación → fatiga; rotá pares.
- *Trampa:* DoD vacío = falso progreso. → Hacela real: build verde + tests + deployable.
- *Trampa:* querer automatizar todo de golpe → empezar por lo mínimo viable (pipeline + tests PR).

- *Trampa:* management espera entregas rápidas sin invertir en prácticas técnicas → educar stakeholders y mostrar ROI (menos bugs, despliegues más confiables). (Scrum.org)

Ejemplo de sprint (flujo integrado Scrum + XP)

1. **Sprint Planning** — definir 3 historias pequeñas, cada una con criterios de aceptación y DoD que incluye tests y despliegue a staging.
2. **Desarrollo** — TDD para la lógica crítica; pair programming en la primera implementación; commits pequeños.
3. **CI** — cada PR dispara pipeline: lint → tests → build.
4. **Review & Merge** — code review + aprobación.
5. **Deploy** — despliegue automático a staging; demo en Review.
6. **Retro** — ajustar prácticas (p.ej. hacer más pair o mejorar flujos de test).

Herramientas recomendadas (rápido)

- **CI/CD**: GitHub Actions, GitLab CI, Jenkins, CircleCI.
- **TEST**: frameworks de unit/integration según stack (JUnit, NUnit, pytest, jest...).
- **Code review**: PRs en GitHub/GitLab + linters.
- **Observability**: Sentry, Prometheus, Grafana para errores y métricas.

10) Checklist rápido para el equipo (usa esto cada sprint)

☐ ¿La DoD incluye tests y build verde?
☐ ¿Cada PR tiene pipeline que pasa?
☐ ¿Se aplicó TDD en al menos una story crítica?
☐ ¿Se usó pair programming en una tarea para transferir conocimiento?
☐ ¿Se deployó a staging y se hizo demo en review?
☐ ¿La retro definió al menos 1 acción para mejorar prácticas técnicas?

11) Mi opinión honesta (fuerte, como pediste)

Si tu equipo tiene problemas de calidad o deuda técnica, **Scrum sin XP es humo bonito**: entregás rápido pero con trompazos. XP te obliga a hacer el trabajo técnico que evita la bancarrota del código. La mejor estrategia es pura: **Scrum para organizar → XP para construir**. Si no vas a invertir tiempo en automatizar tests y CI, no esperes acelerar sin romper cosas. Punto. (Scrum.org)

Si querés, te lo convierto en:

- Un **one-pager** imprimible (resumen ejecutivo + checklist).
- Un **plan de 3 sprints** con tareas concretas para implantar TDD + CI.
- O un **script** con comandos para pipeline CI básico en GitHub Actions (si me decís el stack: Node/.NET/Java/Python).

Te dejo la fuente principal que revisé (explica cómo Scrum y XP se complementan y por qué aplicar prácticas XP dentro de Scrum tiene sentido). (Scrum.org)

Listo. ¿Querés que arme el plan de 3 sprints para tu equipo y lo deje ready-to-run? Te lo hago en modo guerrilla: práctico y sin paja.