Alejandro Fernández García - Ignacio Fernández Suárez – Leonardo Carone Menes

Resumen del Podcast – Sam Taggart y Nicolas Carlo: "Legacy Code First Aid Kit"

1. ¿Qué es Código Heredado?

Nicolas Carlo redefine el "legacy code" no como código viejo, sino como “código valioso que da miedo cambiar”. Este miedo viene de la falta de tests, alta complejidad o poco conocimiento del sistema. A pesar del temor, ese código genera valor y está en producción, por lo que mantenerlo es crucial.

2. Refactorizar, Reescribir o Parchear

- Reescribir completamente es peligroso y rara vez recomendable. Solo se justifica si el alcance es pequeño o el sistema es nuevo.
- Refactorizar de forma iterativa es la estrategia sugerida: extraer pequeños módulos, testear, desplegar gradualmente.
- Parchear (“cinta adhesiva”) se justifica cuando no hay redes de seguridad (tests, monitoreo, logs). Los cambios deben ser mínimos y seguros.

3. Riesgos de Reescribir

Reescribir desde cero puede fallar por:

- Subestimar la complejidad real del sistema.
- No comprender completamente la lógica existente.
- Perder conocimiento tácito durante la migración.

Ejemplo: el gobierno de Quebec fracasó al reescribir su sistema de vehículos, generando retrasos, sobrecostos y mala prensa.

4. Gerencia y Refactorización

Los desarrolladores suelen quejarse de no tener tiempo para refactorizar, pero el problema real es:

1. Falta de un caso de negocio claro: se habla con jerga técnica, no en términos de SLAs, costos o clientes.
2. Cultura que invisibiliza el mantenimiento: se valora solo lo “nuevo”.

5. Cultura y Buenas Prácticas

- Reescribir no es desperdicio, es señal de aprendizaje.
- Aplicar la Regla Scout: siempre dejar el código un poco mejor.
- Establecer “semanas de sostenibilidad” para reducir deuda técnica.
- Involucrar al equipo técnico en decisiones de producto desde el principio.

6. Análisis de Comportamiento

Introducido por Adam Tornhill:

- Usa Git + análisis estático para detectar Hotspots (archivos críticos).
- Permite priorizar refactorizaciones, descubrir acoplamientos y expertos internos.

- Herramienta sugerida: CodeScene.

7. Herramientas y Técnicas del Libro

- Lenguaje-agnóstico.
- Técnicas clave:
    - Hotspots.
    - Inversión de dependencias.
    - Refactorización asistida por el IDE.
    - Nombrado proceso iterativo (mejorar nombres con el tiempo).

8. IA y Código Heredado

La IA puede:

- Generar tests.
- Sugerir refactorizaciones.

Riesgos:

- Sin tests sólidos, puede introducir errores o regresiones.
- La validación humana sigue siendo clave.

9. Método Mikado

Técnica para tareas grandes y complejas:

1. Definir objetivo.
2. Intentar implementarlo.
3. Revertir si falla.
4. Dibujar grafo de dependencias.
5. Resolver subtareas hoja con commits pequeños.

10. Nuevas Herramientas: Moldable Development

Ejemplo: Glamorous Toolkit (GT)

- Permite visualizar sistemas complejos.
- Requiere aprendizaje y está basado en Pharo.
- Potente para sistemas críticos, pero con curva de entrada alta.