Proyectos de gran tamaño: los límites del Extreme Programming

NTE, s.a. / Joan Bosch

Prejuicios acerca de XP

- XP se acostumbra a asociar a:
  - Desarrollo rápido e informal.
  - Análisis y diseño “sobre la marcha”.
  - Proyectos pequeños poco sofisticados.
  - Lanzarse a escribir código cuanto antes mejor (“primero dispara y después pregunta”).
  - Programación ATP/ATM (“Aquí Te Pillo, Aquí Te Mato”).

Qué es NTE (Nuevas Tecnologías Espaciales)

- NTE empezó como una Ingeniería dedicada al mercado aeroespacial (ESA y NASA), y más tarde abrió una división de SW.
- La división de SW se dedica al desarrollo de aplicaciones para el mercado médico.

La necesidades de NTE

- El proceso de desarrollo de SW de NTE, orientado al desarrollo de aplicaciones médicas, debe ser compatible con:
  - Alta fiabilidad y robustez.
  - Fácil mantenibilidad.
  - Flexibilidad frente a cambios de requerimientos

El proyecto TOP

- En 1.999 NTE recibe el encargo de hacer el SW de control de un analizador de sangre de alto rendimiento para una empresa norteamericana.
- Nuestro punto de partida era un HW ya construido y varios miles de requerimientos en forma de sentencias atómicas.
- Dos años más tarde el SW está a punto para el Beta Testing sin errores importantes.

NTE i XP

- Las dos temas de esta exposición son:
  - ¿Cuáles son las técnicas de desarrollo de SW en NTE?
  - ¿Son estas técnicas compatibles con XP?

Técnicas de análisis de requerimientos (I)

- Identificación de tipologías de usuarios.
- Redacción de casos de uso para las áreas funcionales más importantes.
- Test de usabilidad para las áreas funcionales que concentran mayor tiempo de funcionamiento.
- Modelo conceptual completo del mundo del usuario.
- Diccionario de datos.
- Diccionario de pantallas.

Técnicas de análisis de requerimientos (y II)

- Después del análisis funcional, reescribir los requerimientos en forma de sentencias atómicas trazables.
- Establecimiento de un panel de control de cambios.
- Elección de un “usuario clave” como “product manager”.
- “El desarrollador propone y el cliente dispone”

Técnicas de diseño

- Estrecho mapeo entre conceptos de usuario y clases de implementación.
- Revisiones públicas de diseño.
- Agrupación de las clases en grupos de trabajo o “módulos”.
- Definición temprana de las interfaces, empezando por la definición de datos a exportar.

Técnicas de desarrollo (I)

- Planificación temporal monitorizada por característica funcional implementada y terminada.
- Test unitario a nivel de subsistema.
- Integración continua.
- Uso de herramientas de control de cobertura.

Técnicas de desarrollo (II) – Test unitario.

- Todo subsistema debe tener su test unitario.
- El test unitario debe ser:
  - Automático
  - Regresivo
  - Incremental
  - “White Box”

Técnicas de desarrollo (III) – Test unitario

- Ventajas del test unitario
  - Detección temprana de errores.
  - Evita la introducción de errores en la funcionalidad anteriormente implementada.
  - Independencia entre desarrolladores.
  - Facilita la transferencia de módulos de un desarrollador a otro.
  - Facilita el uso de herramientas de cobertura o análisis estático.

Técnicas de desarrollo (IV) – Integración continua

- En todo momento la aplicación ha de compilar, linkar y ejecutar un pequeño test básico.
- Ventajas de la integración continua:
  - Anticipa la detección de errores de integración.
  - Permite a los desarrolladores hacer un mini-test funcional de la funcionalidad implementada.
  - Permite al Jefe de Proyecto monitorizar el estado final de las tareas encargadas.

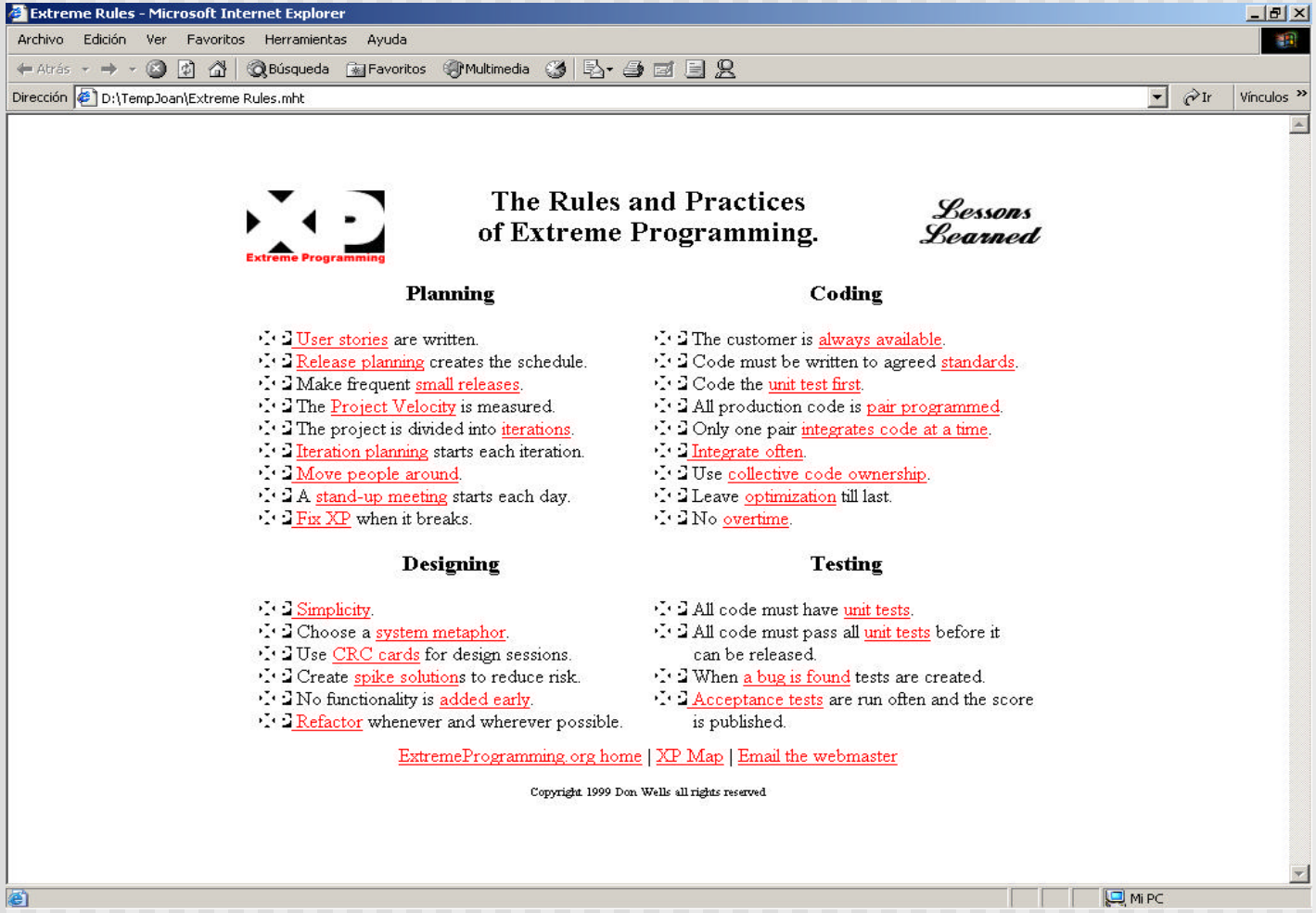
Técnicas de desarrollo (y V) –Herramientas de cobertura

- ¿Qué nos aporta una herramienta de cobertura?:
  - Da idea del grado de exhaustividad del test unitario.
  - Detecta código “muerto”.
  - Detecta un mal análisis de requerimientos al encontrar código que es imposible ejecutar.
  - Ayuda a detectar el exceso de carga de trabajo en un desarrollador.

Técnicas de aseguramiento de la calidad

- Revisiones de especificaciones y diseño por un equipo independiente de QA.
- Revisiones de código por un equipo independiente de QA.
- Automatización del test de sistema.
- Test de calificación diario.

Las reglas del XP



NTE vs XP: reglas incluidas(I)

- Pequeñas entregas frecuentes.
- Proyecto dividido en iteraciones.
- Se hace un plan de iteración con cada nueva iteración.
- Diseño tan simple como sea posible.
- Crear modelos para minimizar riesgo.
- Reaprovechar código siempre que sea posible.
- El cliente siempre ha de estar accesible.

NTE vs XP: reglas incluidas (y II)

- El código ha de cumplir estándares.
- Integrar a menudo.
- Dejar la optimización para el final.
- Todo código tiene su test unitario.
- Todo código ha de pasar su test unitario antes de ser entregado.
- Añadir un nuevo caso de test a cada error encontrado.
- El test de aceptación se ejecuta a menudo.

NTE vs XP: reglas compatibles (I)

- Documentar las necesidades de los usuarios (*User stories*)
- Hacer la planificación a partir de la entrega.
- Medir la velocidad del proyecto.
- Hacer una reunión “de a pie” al empezar el día.
- Cambiar el proceso de desarrollo si no funciona.
- Elegir una metáfora del sistema.
- Usar tarjetas CRC para el diseño.

NTE vs XP: reglas compatibles (y II)

- No añadir funcionalidad tempranamente.
- Codificar el test unitario antes que la implementación.
- Solamente un par de desarrolladores integra a la vez.
- No sobrecargar de trabajo a los desarrolladores.

NTE vs XP: reglas NO compatibles

- Promover la movilidad de los desarrolladores entre proyectos.
- Codificar la misma pieza por dos desarrolladores a la vez (*pair-programming*).
- Propiedad colectiva del código.

Análisis reglas no compatibles

- La alta especialización de los proyectos de NTE va en contra de:
  - Promover la movilidad entre desarrolladores.
  - Propiedad colectiva del código.
- La codificación por pares jamás ha sido probada en NTE debido a la cultura propia de la empresa.

Conclusiones

- El modelo de desarrollo de SW de NTE es altamente compatible con XP.
- Algunos ajustes son necesarios debido a la cultura empresarial de NTE y al alto nivel de especialización de sus aplicaciones.