

Ponencia

Trabajando con Programación Extrema – Experiencias Reales

Autor: Miguel Jaque Barbero (mjaque@ilkebenson.com)

Palabras Clave: software, proyectos, metodología, programación extrema, prácticas, experiencia.

Los Problemas del Desarrollo de Software

¿Qué está pasando?

El éxito de un desarrollo de software, como cualquier otro proyecto, se mide en base a cuatro parámetros: plazo, coste, funcionalidad y calidad. Sin embargo, frecuentemente los proyectos de software exceden su plazo y su coste, el sistema obtenido suele carecer de la funcionalidad necesaria y presenta un bajo nivel de calidad (Ref. Chaos Report – Standish Group 1995). Esto no ocurre en otras ingenierías. ¿Porqué en los proyectos de software aceptamos como normales unos resultados tan malos? ¿Cuál es la causa de este fracaso?

¿Cuál es la causa?

El motivo principal es el riesgo. Los proyectos de desarrollo de software afrontan mayores niveles de riesgo que los de otras ingenierías (construcción civil, aeronáutica, ingeniería naval...). En un proyecto de software, el riesgo aparece en forma de errores y de cambios.

Los primeros son debidos a la propia complejidad de la programación. Esta es una ciencia nueva y altamente cambiante. Los programadores no pueden adquirir experiencia por la simple razón de que, en un plazo no superior a dos años, cambiarán de tecnología. Y la falta de experiencia favorece la aparición de errores.

La segunda fuente de riesgo son los cambios. Estos se deben a la evolución del entorno del usuario, su mercado, su estrategia, la legislación, las condiciones laborales... que inevitablemente deben verse reflejadas en las aplicaciones de soporte para su negocio. Hoy, en muchos sectores, esta evolución es tan rápida que los cambios se producen durante el propio desarrollo del proyecto.

¿Cuál es la solución?

En este contexto de riesgos inevitables pero al mismo tiempo altamente costosos, los ingenieros de desarrollo de software siempre han buscado la experiencia de gestión que no tienen, en otras ingenierías. Así surgieron las metodologías tradicionales del desarrollo de software, basadas en un modelo secuencial de actividades (planteamiento – diseño – planificación – ejecución – revisión) típico de la ingeniería civil.

Y, sin embargo, es este mismo modelo, con su enfoque secuencial, el que contribuye a

maximizar los efectos negativos de los errores y los cambios, haciendo que su detección se retrase en el tiempo y contribuyendo a incrementar los costes de corrección y codificación que tienen asociados.

La Programación Extrema

Frente a las metodologías tradicionales, Kent Beck planteó en 1999 la Programación Extrema (Ref. Extreme Programming Explained – Addison Wesley 1999). Basándose en cuatro valores fundamentales (comunicación, sencillez, realimentación y valentía), Kent Beck seleccionó y organizó las prácticas más convenientes para el desarrollo de software. Para ello, propuso la radicalización de su uso:

“Si diseñar es bueno, diseñemos todo el tiempo.

Si probar es bueno, probemos todo el tiempo...”.

Como resultado de su planteamiento, apareció una de las primeras metodologías consideradas ágiles, que aborda las actividades de programación de forma paralela, no secuencial.

Los Valores

La Programación Extrema busca prácticas que maximicen los siguientes valores:

Comunicación

Muchos de los problemas de los proyectos vienen de la falta de comunicación. El cliente no dijo algo importante al programador, o este no le hizo la pregunta adecuada que hubiera puesto de manifiesto el nuevo cambio, o alguien no comunicó un cambio importante en la estructura de clases... Y, sobre todo, sin una buena comunicación, el programador no entenderá el valor que para el cliente tiene un cambio ni este entenderá que el programador pueda cometer errores.

Por eso, la Programación Extrema selecciona prácticas que obligan a la comunicación. La programación en parejas obliga a los programadores a comunicarse, el juego de la planificación obliga al jefe de proyecto a hablar con los clientes, las estimaciones obligan a los programadores a hablar con el jefe de proyecto...

Sencillez

La Programación Extrema apuesta por desarrollar hoy lo mínimo necesario y pagar algo más mañana, si es que finalmente se necesita más. La sencillez es muy difícil, porque obliga a ignorar las necesidades de mañana y centrarse exclusivamente en las necesidades más inmediatas.

Pero merece la pena. Un sistema más sencillo requiere menos comunicación y, por lo tanto, será más fácil comunicarlo completamente. Un sistema sencillo se puede modificar más fácilmente y es menos susceptible a errores. En un sistema sencillo es mucho más agradable

trabajar y simplificarlo más siempre es reconfortante.

Realimentación

La realimentación es el tratamiento contra el optimismo. Es frecuente que, por ejemplo, las mediciones del avance del proyecto se midan “de forma subjetiva”. En XP no. Si quieres saber algo, se lo preguntas al sistema, ¿cuántas pruebas de aceptación pasa? ¿cuántas historias se han validado?

La realimentación debe ser rápida, para que los cambios y los errores se detecten y corrijan cuanto antes, con el menor coste.

La realimentación funciona a diferentes escalas de tiempo. Los programadores detectan sus errores con las pruebas unitarias en cuestión de horas. Los clientes detectan la validez de sus historias con la estimación de los programadores, inmediatamente. Y conocen el estado del sistema en cuestión de días o semanas, conforme van pasándose las pruebas de validación.

Además, la realimentación potencia la comunicación y la sencillez. Un caso de prueba que demuestre un fallo ahorrará horas de discusiones. Y cuanto más sencillo sea un sistema, más fácil será probarlo.

Valentía

Tirar el 80% del código del proyecto a cuatro semanas de la conclusión para conseguir un sistema más sencillo, requiere valentía.

Medir honestamente el avance del proyecto y comunicar abiertamente los errores y los cambios, requiere valentía.

Pero, si no te enfrentas a estos problemas con valor, otros lo harán, y estarás fuera.

La Programación Extrema te indica el camino, pero no quita los lobos. Ese es tu trabajo.

Los Principios

Pero estos valores son demasiado generales, necesitamos algo más concreto para poder identificar correctamente las prácticas que necesitamos. Para eso están los principios.

Hay cinco principios fundamentales y varios más adicionales. Los fundamentales son:

- **Realimentación Rápida:** De esta forma se maximiza el aprendizaje y se reduce el coste de los cambios y de los errores.
- **Asumir la Sencillez:** Para todo problema complejo hay siempre una solución sencilla. Tenemos que trabajar para encontrarla, empezando por aceptar que existe.
- **Cambio Incremental:** Nada de grandes cambios, no funcionan. La diferencia se consigue mediante una sucesión de pequeños cambios que harán la diferencia.
- **Aceptar el Cambio:** Optaremos por las estrategias de desarrollo que nos permitan posponer las decisiones al máximo. De esta forma, el cliente no tendrá que optar ni invertir hasta el último momento, con la máxima información y el menor coste.

- **Trabajar con Calidad:** Los trabajadores tienen derecho a hacer un trabajo de calidad, del que puedan sentirse orgullosos. Nada de chapuzas.

Y, los principios adicionales son:

- Enseñar a Aprender.
- Pequeña Inversión Inicial.
- Jugar a Ganar.
- Experimentos Concretos.
- Comunicación Abierta y Sincera.
- Trabajar con los Instintos de las Personas.
- Aceptar la Responsabilidad.
- Adaptación Particular.
- Ir Ligero de Equipaje.
- Medidas Honestas.

Estos son los principios que nos ayudan a elegir las mejores prácticas.

Las Prácticas

Las prácticas son las técnicas concretas y detallada que tenemos que utilizar. Algunas son llamativas, como la programación en parejas, la propiedad colectiva e incluso las prohibiciones de hacer horas extraordinarias o de posponer una fecha de entrega.

No podemos entrar en una descripción detallada de todas ellas, pero al menos vamos a enumerarlas:

- Juego de Planificación
- Versiones Pequeñas
- Metáfora de Desarrollo
- Diseño Sencillo
- Pruebas Automáticas
- Recodificación
- Programación en Parejas
- Propiedad Colectiva
- Integración Continua
- 40 Horas Semanales
- Cliente in-situ
- Estándares de Codificación

Lo más importante es que todas ellas se apoyan entre sí.

Por ejemplo, utilizar estándares de programación es mucho más fácil y efectivo si ya se programa en parejas, porque la comunicación sobre las reglas de estilo es mucho más fluida. Al mismo tiempo, la recodificación (refactoring) es casi imposible sin las pruebas unitarias. Sin ellas, no podríamos saber si la recodificación está concluida o no. Y la integración continua no puede aplicarse sin la propiedad colectiva del código, porque se bloquearían las correcciones del código y el avance del proyecto.

Experiencia Real con Programación Extrema

ILKE BENSON se propuso adoptar esta metodología en el año 2002. Desde entonces nos hemos enfrentado, y casi siempre superado, a un montón de problemas. ¿Cómo enfocamos nuestro cambio de metodología? ¿Qué problemas afrontamos?

En primer lugar, la decisión de probar la Programación Extrema fue planteada por la Dirección. Así que nos evitamos uno de los mayores problemas (“¿*Dos programadores con un teclado? Eso supone la mitad de productividad*”). Y además, nos planteamos la transición poco a poco, adoptando las prácticas de forma progresiva.

Pero, la Programación Extrema, tal y como la plantea Beck en su libro, tiene muchos problemas. Algunos son pequeños y fácilmente superables, consecuencia de una nueva forma de trabajo y de la resistencia al cambio de unos personajes “curiosos”; los programadores.

Pero otro, que abordaremos después, es “casi infranqueable”; ¡La Programación Extrema ignora completamente el presupuesto del proyecto!

Pero, si tiene tantos problemas, ¿porqué intentarlo?

Beneficios de la Programación Extrema

Y sin embargo, a pesar de los problemas que hay que superar, merece la pena intentarlo.

El premio es grande: satisfacer completamente al cliente entregándole toda la funcionalidad que necesita, acabar los proyectos a tiempo y dentro del presupuesto, y trabajar de una forma mucho más humana, más divertida y más gratificante.

Satisfacción del Cliente

La Programación Extrema consigue la máxima satisfacción del cliente porque le permite cambiar los requerimientos incluso del código ya desarrollado. De esta forma, el resultado final incluye cualquier idea nueva del cliente, incluso las que ni siquiera podía imaginar al inicio del proyecto.

Cumplimiento de Plazos

Y termina en el plazo previsto. Claro que para ello elimina funcionalidad si es necesario. Pero este recorte se realiza de acuerdo con el cliente, posponiendo la funcionalidad de menor valor y poniendo en explotación antes la que le resulta más rentable. Lo del presupuesto es otro cantar que abordaremos enseguida.

Calidad en el Trabajo

Y para los programadores es el paraíso. En lugar de trabajar aislados, trabajan en parejas, en constante comunicación con el resto del equipo, incluyendo el cliente. Abandonan el estrés de las fechas de entrega porque pueden posponer la funcionalidad. No hacen horas extras porque está prohibido. Y, sobre todo, ¡no tienen que hacer documentación!

Puede que el camino esté plagado de problemas, pero merece la pena intentarlo.

Pequeños problemas internos

Hay dos problemas que surgen de la novedad que para los programadores supone la propia Programación Extrema.

En primer lugar, su Resistencia al Cambio.

Resistencia al Cambio

Adoptar una nueva práctica obliga a abandonar la anterior y a aprender una nueva forma de trabajo. Esto implica ganas de mejorar y ganas de aprender. Pero, en muchos casos, la respuesta es escéptica (“*Eso no va a funcionar*”).

Si unimos a esto que las prácticas en Programación Extrema están diseñadas para apoyarse unas a otras, frecuentemente nos encontramos que la adopción de una nueva práctica conlleva un esfuerzo de comunicación y aprendizaje que no compensa los beneficios obtenidos inicialmente. Dicho de otra forma, las nuevas prácticas no compensan hasta que no se apoyan con otras nuevas. Es una pescadilla que se muerde la cola.

Por eso es muy importante empezar adoptando las prácticas más rentables, más fáciles y más divertidas, como la programación por parejas o la prohibición de hacer horas extras, para reducir así la resistencia al cambio.

Y en segundo lugar, está el problema de “La Pasión por Programar”.

Pasión por Programar

A los programadores les apasiona programar. Desgraciadamente esto suele implicar que no les gusta comunicarse, que prefieren los retos complicados a buscar soluciones sencillas y que tienen aversión a perder el tiempo haciendo pruebas, en lugar de programar o mejorar lo que ya han programado.

La Programación Extrema requiere pensar antes que programar. Y un programador, apasionado por su trabajo, ante un nuevo reto, suele lanzarse al teclado antes de pararse a pensar qué tiene que hacer y cómo.

Afortunadamente, la solución también está en esta metodología (“Trabaja con los instintos de las personas, no contra ellos”). Se trata de crear la cultura en la empresa de que el buen programador no es aquél que resuelve los problemas más complicados, sino quien sabe evitarlos. Transmitir que son mejores programadores quienes saben entender a su cliente, más que quienes saben entender una librería compleja de programación.

Cambiar la cultura es un trabajo lento, pero ¿hay algún otro?

El gran problema del presupuesto

Los dos problemas que hemos visto, la Resistencia al Cambio y la Pasión por Programar, tiene soluciones “fáciles”. Pero hay otro problema mucho mayor que no podemos seguir esquivando: ¡La Programación Extrema ignora el presupuesto del proyecto!

Realmente no es que lo ignore, simplemente lo sacrifica en favor de la satisfacción del cliente. Por eso da tan buenos resultados en proyectos *in-company*.

Es decir, cuando un departamento de informática de una organización adopta la Programación Extrema, el resto de departamentos, que realmente actúan como sus clientes, tienen garantizada la satisfacción en los desarrollos que aborden. Los programadores de su empresa aceptarán cualquier cambio, en cualquier momento, que ellos le soliciten. Simplemente les informarán en qué fecha podrán tenerlo listo. E incluso en eso, el cliente podrá negociar la funcionalidad a incluir para alcanzar las fechas necesarias.

Pero no todos los proyectos son así. ¿Qué pasa con los proveedores externos? ¿Pueden las empresas de desarrollo de software, esas que trabajamos para clientes externos, adoptar la Programación Extrema?

El problema está en el presupuesto. Casi todos los clientes quieren presupuestos cerrados. Es decir, acuerdos previos de funcionalidad y plazo a cambio de coste (“*Vamos a hacer toda esta funcionalidad, en este plazo y por este precio*”). Pero entonces... ¿qué pasa con los cambios?

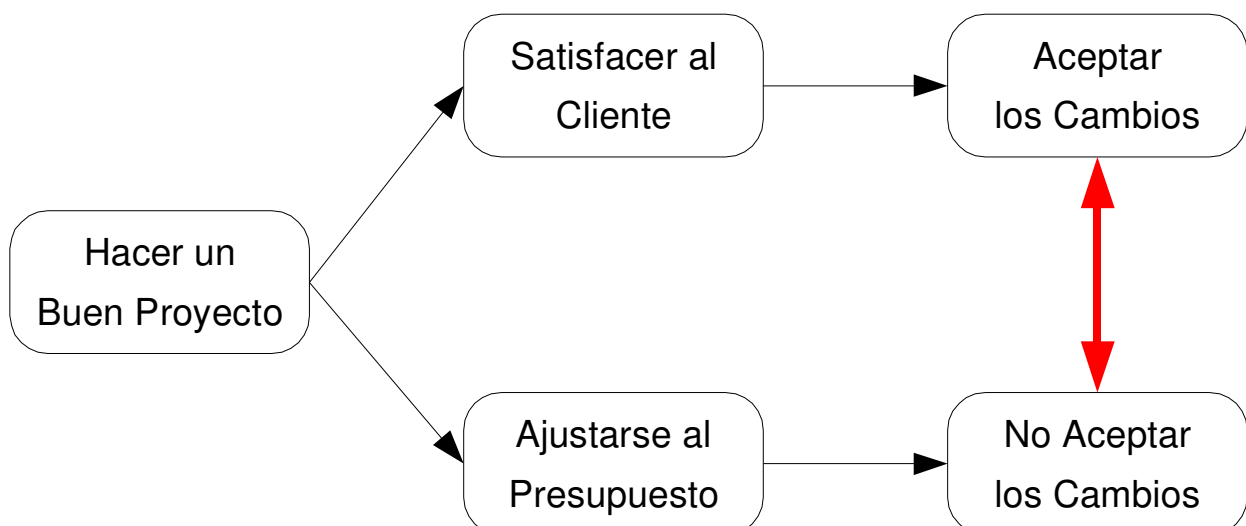
Teoría de las Limitaciones

Estamos ante un conflicto que es ignorado por la Programación Extrema. Para un “programador extremo” hacer un buen proyecto supone satisfacer al cliente. Y para satisfacer al cliente tiene que aceptar sus cambios. La empresa para la que trabaja le garantiza su nómina, su presupuesto.

Para un “programador externo”, hacer un buen proyecto no solo implica satisfacer al cliente y, por lo tanto, aceptar sus cambios. También supone cumplir con el presupuesto, y para ello, ¡no debe aceptar cambios!

Necesitamos nuevas herramientas para resolver este problema. Y las podemos encontrar en la Teoría de las Limitaciones (TOC - Theory of Constraints) enunciada por Eliyahu M. Goldratt en “No es Cuestión de Suerte”.

Gráficamente, el conflicto al que nos enfrentamos se expresa en el siguiente gráfico:



Los recuadros muestran afirmaciones y las flechas señalan las implicaciones entre ellas.

Debe leerse así: “Hacer un Buen Proyecto implica Satisfacer al Cliente”. “Hacer un Buen Proyecto implica Ajustarse al Presupuesto”. “Satisfacer el Cliente implica Aceptar los Cambios”. “Ajustarse al Presupuesto implica No Aceptar los Cambios”.

La flecha roja simplemente señala el conflicto que debemos resolver.

TOC nos dice que desechemos las soluciones de compromiso. “Aceptar algunos cambios” no es resolver el conflicto. Y todo conflicto tiene solución. Para solucionarlo, basta con buscar la flecha que menos nos guste, aquella con la que no nos sintamos a gusto y cuestionar la suposiciones que implica.

Evitando los Cambios

Nosotros, en [ILKE BENSON](#), nos concentramos en la flecha superior “Satisfacer al Cliente implica Aceptar los Cambios”. Y durante un año trabajamos en evitar que el cliente tuviera que proponer cambios.

Para ello, utilizamos técnicas de captura de requerimientos. Las incluimos en cada iteración del proyecto y buscábamos ayudar al cliente a definir lo que realmente necesitaba. De esta forma, conseguimos reducir los cambios, pero no eliminarlos.

A pesar de lo ajustado de los Diseños Detallados, en ocasiones el entorno del cliente nos obligaba a cambiar (modificaciones legales durante el desarrollo del proyecto, cambios en la organización...) Y en otras ocasiones, los errores en la limitación del ámbito del proyecto, aquella con la que se establece el presupuesto de desarrollo, traían nuevos cambios... Estábamos trabajando contra uno de los principios fundamentales de los proyectos de software, ¡los cambios son inevitables!

No despreciamos lo que hemos conseguido, los cambios se han reducido. Pero tenemos que trabajar con la flecha inferior “Para Ajustarnos al Presupuesto tenemos que Rechazar los Cambios”.

¿Cómo podemos cuestionarnos esta suposición?

El Presupuesto y Los Cambios

Si seguimos fielmente a TOC, y puesto que la suposición superior ha resultado más sólida de lo esperado, la suposición inferior debe tener algún punto débil.

Para ello, nos hemos planteado la siguiente estrategia.

En primer lugar, convencemos al cliente de que su proyecto va a sufrir cambios respecto al diseño inicial. Eso no es difícil, ellos lo saben. Y después, le ofrecemos tres mecanismos para gestionar esos cambios:

- **Un margen de seguridad.** Es decir, una bolsa de presupuesto y plazo adicional al previsto que iremos consumiendo conforme surjan los cambios.

- **Negociación de Funcionalidad.** En lugar de consumir el margen, o si este ya ha sido liquidado, podemos intercambiar la nueva funcionalidad que conlleva un cambio con parte de la funcionalidad prevista que todavía no se hubiera desarrollado. Básicamente consiste en cambiar lo prioritario por lo menos prioritario. El proyecto terminará con menos funcionalidad de la esperada, pero con la funcionalidad más importante para el cliente.
- **Incremento del Presupuesto.** Si un cambio no puede ser asumido con el margen de seguridad, ni hay posibilidad de negociar un cambio de funcionalidad, entonces tendremos que incrementar el presupuesto y el plazo de ejecución.

Todavía no podemos determinar si esta nueva aproximación al problema lo soluciona o no. Acabamos de ponerla en marcha. Pero parece la única posibilidad de Aceptar Cambios y Ajustarnos al Presupuesto.

Puede que nunca lleguemos a aplicar completamente la Programación Extrema o puede que sí. Pero lo que sí es cierto es que el camino que recorreremos al intentarlo no está aportando resultados muy valiosos.

Currículum del Autor

Miguel Jaque Barbero (mjaque@ilkebenson.com)

Tel:(+34)609 164 628

Soy Ingeniero Superior de Telecomunicación por la Universidad Politécnica de Madrid y Máster en Administración de Empresas por el Instituto de Empresa de Madrid.

En 1999 creé mi propia empresa, [ILKE BENSON](#), dedicada al desarrollo de aplicaciones y servicios informáticos basados en Software Libre. Actualmente desarrollamos sistemas y aplicaciones informáticas para Bancos, Administraciones Públicas y Grandes Empresas a nivel nacional. Hoy, [ILKE BENSON](#) lo formamos 12 personas, distribuidas en dos centros situados en Badajoz y Cáceres. Entre otros, participamos en proyectos como gnuLinEx (el Sistema Operativo de la Junta de Extremadura), LíneaCB (el interfaz de Banca por Internet de Caja Badajoz), STAR (Terminal Financiero de Banca Pueyo), SIRVEM (Simulador Virtual de Empresas del Ayuntamiento de Cáceres) y Escalinova (Portal de la Innovación de Fundecyt).

Toda mi carrera profesional se ha desarrollado en el sector del software. Trabajé como Jefe de Proyecto en el Consejo Superior de Investigaciones Científicas, donde Javier me enseñó como tratar a los clientes. Después como Jefe de Soporte Técnico en Advanced Vision Technologies, donde Raimundo, Carlos y Alberto me enseñaron como gestionar una empresa de desarrollo de software. Y representé a varias empresas internacionales en España (Mercury Interactive y Centura Software Corporation) antes de embarcarme con [ILKE BENSON](#).

He publicado dos artículos en la revista DATA.TI: "Una Introducción a la Programación Extrema" y "Conceptos Económicos del Desarrollo de Software". Y algún otro artículo mío ha aparecido en Expansión, ICTNET ("La Adicción a las Bases de Datos"), Computer World e incluso Tecnimap ("Metodologías de Test para el Año 2000").

Soy Profesor Colaborador de la Escuela de Organización Industrial (EOI) e imparto seminarios de Gestión de Proyectos de Software, Software Libre y Organización Técnico-Comercial a empresas y organizaciones.