

San Sebastián, Junio 2008

Buenas prácticas de gestión en empresas de servicios avanzados

Uso de técnicas de metodologías ágiles de
desarrollo con CMMI nivel 2 en un entorno
corporativo de Banca

Versión 1.0



INFORMATION TECHNOLOGY ADVISORS

INTRODUCCIÓN	3
ESCENARIO	4
ORGANIZACIÓN Y GESTIÓN	5
EL EQUIPO	7
DESARROLLO	7
CONCLUSIONES	10



Introducción

Recientemente Biko obtuvo la certificación CMMI nivel 2. Tras un periodo de estudio de los procesos convenientes para el funcionamiento de la organización, estos fueron validados y aprobados mediante el SCAMPI.

Este nivel de la metodología formal se centra en determinadas áreas, como planificación, gestión de requerimientos, métricas, y verificación y validación.

En Biko, CMMI ha servido para uniformizar criterios importantes sobre la gestión de los proyectos, que dada la heterogeneidad de los múltiples proyectos desarrollados en la organización, ha sido un hito muy importante.

Pero CMMI en su nivel 2 no especifica nada sobre las metodologías de desarrollo o gestión del equipo, ni del proceso concreto de creación de software. Es por eso por lo que hemos ido un paso más allá, y hemos buscado técnicas para el mejor control del desarrollo.

Las metodologías ágiles de desarrollo van hacia otro objetivo que las metodologías formales. Se centran *"en los individuos y sus interacciones más que en procesos y herramientas"*¹. Algunas de las más importantes son las basadas en el concepto de *"Lean development"*², u otras más concretas como pueden ser *"Scrum"* y *"XP"*.

¹ <http://agilemanifesto.org/>

² Desarrollo "ligero": Basado en los principios del "lean manufacturing" aplicados al software.



Escenario

Tras un proyecto desarrollado con algunos problemas para un cliente de Banca, se nos plantea hacer un segundo desarrollo. Es entonces cuando buscamos la solución que mejor pueda solucionar los problemas encontrados anteriormente.

Los problemas a los que intentamos dar solución:

- Pérdida de control del proyecto por los desarrolladores. Conocen parte del proyecto, pero a veces repetimos soluciones diferentes para el mismo problema.
- Se intentó entregar por iteraciones, pero estas nunca quedaban definidas, y pronto se perdió el control exacto de las funcionalidades incluidas en cada publicación.
- Las peticiones de cambio generadas por un cliente muy pendiente del desarrollo del proyecto, eran muy numerosas. La gestión de requerimientos mediante una hoja Excel no ayudaba demasiado.
- El cliente quedó medianamente satisfecho con el producto final, pero este quedó con carencias importantes. Además, la etapa de desarrollo fue muy dura tanto para el equipo como para el cliente.



Organización y Gestión

Para el segundo proyecto nos planteamos utilizar metodologías ágiles, para mejorar la eficiencia del equipo. ¿Por qué? Estos son algunos de los puntos que tomamos en cuenta como punto de partida, donde las metodologías ágiles podían ayudarnos más:

- “*Empowerment*” del equipo. Reforzar su capacidad de decisión y confianza.
- Basadas en iteraciones y entregables funcionales:
 - Proporcionando mayor control para el cliente.
 - Y mayor control de las versiones por parte del equipo, sobre qué funcionalidades estaban realizadas y cuáles en desarrollo.
- Además, compatible con nivel 2 de CMMi

Como primera experiencia nos basamos en SCRUM. Pero no la implementamos al 100%, realmente la adaptamos. Así que “no decimos que usamos Scrum”. Pero, ¿por qué lo modificamos?

- El alcance ya estaba definido con el cliente. Los contratos a precio cerrado no son el mejor encaje para las metodologías ágiles.
- Requeríamos un análisis previo, para la validación por el cliente, una fase inicial donde estudiásemos la globalidad del proyecto.



Como este proyecto era la continuación tecnológica de uno anterior, minimizábamos el riesgo, y podíamos extrapolar mejor las conclusiones de la implantación.

- Proyecto tecnológicamente conocido: El primer proyecto fue el que estableció las bases tecnológicas, y salvó los escollos más importantes en esta área. Teníamos mucho trabajo por hacer, para mejorar la plataforma creada, pero consistía en refactorizar elementos.
- Cliente recurrente: conocíamos cómo trabaja, y podíamos adaptar nuestra manera de desarrollar para su satisfacción.



El equipo

El equipo, visto desde el prisma de sus roles tradicionales ha sido: dos desarrolladores, un diseñador, y un analista funcional y un jefe de proyecto. Realmente el equipo que trabajó bajo las premisas de gestión ágil fueron todos excepto la gente de diseño, puesto que trabajaron en momentos más puntuales.

Los perfiles eran bastante distintos en cuanto a experiencia, conjugando dos personas con un año escaso, con otras de casi 10 años en el desarrollo de software.

Desarrollo

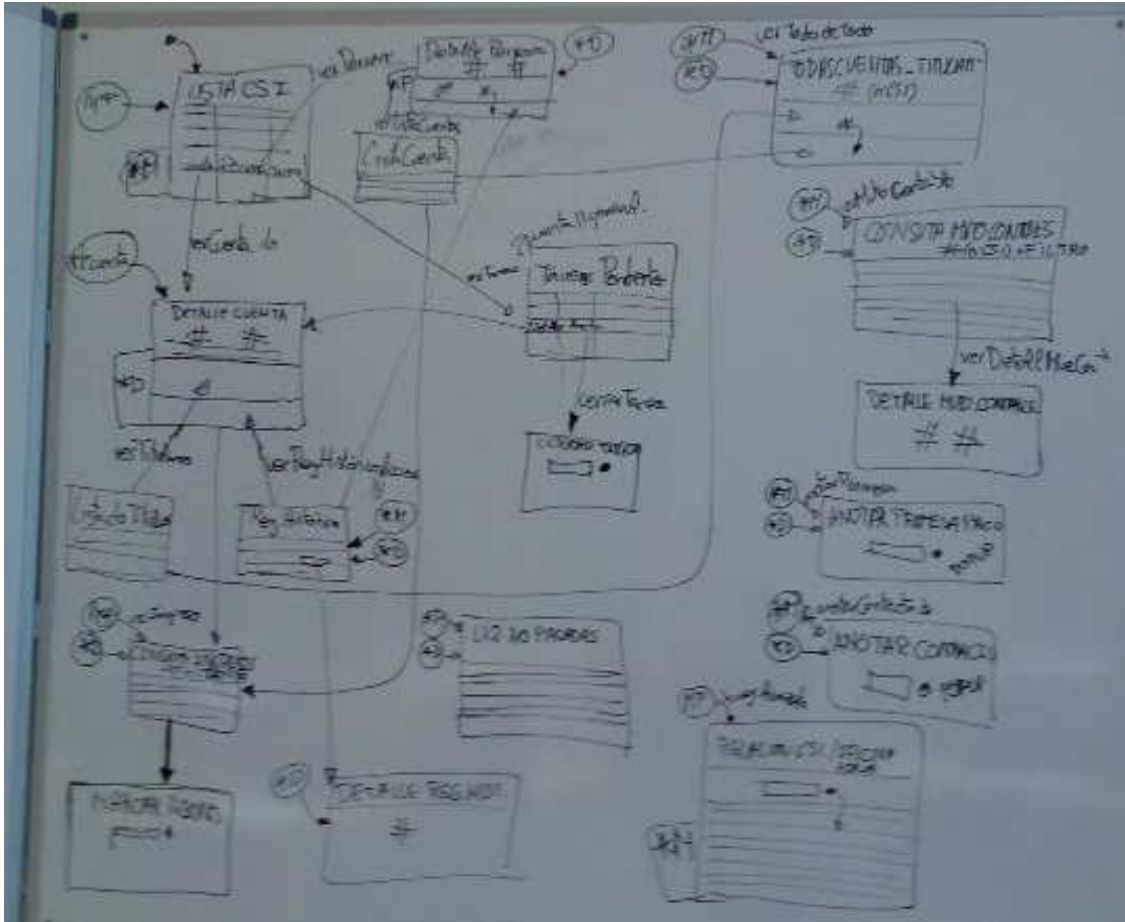
La idea inicial era utilizar Scrum para el desarrollo del proyecto. Sin embargo, echando la vista atrás no podemos decir que lo hayamos utilizado, puesto que hemos hecho modificaciones importantes, y no hemos aplicado todas sus técnicas.

Antes de empezar el ciclo de iteraciones hicimos una fase cero del proyecto:

- **Fase -1:** Preparación del catálogo de requisitos.(Product Backlog)
- **Fase 0:** Análisis funcional, y diseño de la interfaz y su arquitectura. Esta fase fue requerida por el cliente para su validación.
- **Fase N, Iteraciones semanales:** Trabajo definido por el equipo. Cierre con demos de la versión y puesta en desarrollo para el cliente.

Nuestro grado de avance y situación lo llevamos en una pizarra y en una herramienta de gestión de proyectos (JIRA) simultáneamente.





- La pizarra proporciona una inmediatez del trabajo efectuado/restante que da una sensación de control del proyecto muy importante. Es un mapa que se puede ver con solo girar la cabeza, y ino hay que hacer un solo *click*! Scrum recomienda la pizarra de "post-it", pero creo que esto es igual de útil.
- JIRA nos proporciona soporte para asociar a las tareas documentación, partes de horas, comentarios de los desarrolladores, enlaces al código de SVN..., a costa de un mayor gasto de gestión, pero que, una vez habituados a la herramienta, es muy escaso.



Además también nos proporciona el EDT de las tareas actualizado, basándonos en las tareas cerradas/abiertas y sus estimaciones.

También sustituimos la gestión de cambios de requerimientos, pasando del "Excel" a la gestión de tareas identificadas como "Nuevos Requerimientos"

Por cada iteración definíamos qué trabajo íbamos a realizar para publicar (Sprint Backlog) como resultado de esa semana (Sprint).

Cada 15 días, además, realizábamos una reunión de seguimiento con las personas no directamente implicadas: gerente y diseñador.



Conclusiones

Este ha sido un proyecto de introducción de novedades en gestión del equipo de desarrollo de software. Estamos aplicando por ejemplo en otros proyectos una aproximación mucho más estricta de Scrum. Pero he aquí algunas conclusiones que podemos extraer:

- 1) Las iteraciones semanales son demasiado cortas. Este proyecto era de duración reducida, pero unas iteraciones tan cortas introducen mucho "ruido" del cliente demasiado cerca de la siguiente finalización de la iteración.
- 2) No aceptar cambios en el plan de iteración (solo corrección de "bugs" de la iteración anterior, para lo que se deja un tiempo) es una gran idea. Puedes controlar cómo funcionan respecto a tu planificación.
- 3) El equipo controla todas las partes del desarrollo. Las reuniones diarias de 10' son puestas como ejemplo y destacadas por los integrantes del equipo como una maravillosa práctica.
- 4) Los perfiles de menos experiencia se benefician en gran medida de las reuniones diarias, donde se les resuelve el 90% de sus problemas. Nunca se bloquean en un trabajo más de 8 horas sin que el equipo lo sepa.
- 5) El problema de la gestión del proyecto como horas cerradas sigue pendiente. Tienes unas horas de una estimación inicial, que engloban a todas las funcionalidades, y realmente no ves y estimas adecuadamente las tareas hasta el principio de cada iteración.



- 6) La primera iteración o fase 0, es muy interesante, porque proporciona una base común para el trabajo posterior. En este caso no teníamos problema para la solución técnica, pero si no hubiese estado hecha en un proyecto anterior, posiblemente la hubiésemos planteado aquí. De esta manera podemos establecer una base de cómo realizar las cosas, útil especialmente trabajando con desarrolladores más inexpertos.
- 7) Todo el equipo involucrado ha destacado la utilidad de las reuniones diarias, especialmente la gente con menos experiencia.

