CÓMO AGILIZAR LA GESTIÓN DE PROYECTOS DE SOFTWARE

“Hay mejores formas de desarrollar software”

Texto
Javier Garzás, 233 Grados de TI y Universidad Rey Juan Carlos

“Es clave el papel del testing en todas sus dimensiones, la calidad del código y controlar la llamada deuda técnica”

La gestión del desarrollo software del Healthcare.gov, que implementa la reforma sanitaria de Obama, ha sido el último gran desastre en lo que se refiere a gestión de proyectos de software y uno de los mayores de la historia. Un presupuesto de 400 millones de dólares, con demoras interminables, errores y que al poco de salir a producción… se cayó.

Numerosos medios y expertos pusieron el principal foco del desastre en cómo se gestionó el desarrollo: siguiendo el pesado, tradicional y anticuado ciclo de vida en cascada, una de las mayores críticas que recibió el proyecto.

Esto llevó, como apuntan muchos medios(1), a que, por ejemplo, las pruebas del Healthcare.gov se realizaran solo las dos últimas semanas del proyecto. Incluso el NY Times(2) llegó a criticar que el desastre se hubiese evitado con “la adopción de prácticas modernas de desarrollo de software incremental, como la agilidad”.

Fue el Manifiesto Ágil, en 2001, quien introdujo por primera vez el término ‘ágil’ en el contexto del desarrollo de software. Un conciso conjunto de valores que comienza diciendo que “estamos descubriendo mejores formas de desarrollar software”(3), los llamados métodos de desarrollo de software ligero, que nacieron como respuesta a prácticas y ciclos de vida software pesados; en particular el ciclo de vida en cascada, que emula los métodos de construcción de cosas físicas, la industrialización, adaptándolos al mundo del software.

La maldición de ‘industrializar’ el desarrollo

Prácticamente desde su nacimiento, la primera conferencia sobre ingeniería software, aquella mítica organizada por la OTAN en el 68 (que popularizó términos como ‘fábricas de software’ o la propia ‘ingeniería de software’), empujó a la comunidad a imitar los métodos tradicionales de gestión (‘software designers are in a similar position to architects and civil engineers’, citaban las actas), cuya máxima expresión ha sido el ciclo de vida en cascada.

El ciclo de vida en cascada deriva de las industrias de productos físicos, de la construcción, donde detectar un fallo en el producto ya construido es extremadamente costoso de arreglar (cambiar, por ejemplo, la posición de una columna en un edificio).

Y de ahí que la visión clásica de la gestión, heredada por muchos proyectos, pretenda lograr requisitos, diseños o planos de un alto nivel de detalle, que una vez terminados no se cambien (requisitos cerrados, proyectos cerrados, plazos cerrados, etc., el rechazo al cambio) con fases (requisitos, análisis, diseño, etc.) que se realizan (en teoría) linealmente, una única vez, y cuyo resultado se pretende inamovible.

La historia nos ha demostrado que es muy difícil especificar en una única, inamovible y primera fase los requisitos, es hasta contraproducente para los negocios evitar el cambio en los mismos e, igualmente, resulta complicado cerrar en una única fase de diseño todas las cuestiones a tratar en la programación.

El software, por su naturaleza, y si se construye correctamente, es fácil de modificar, de añadirle funcionalidades evolutivamente, siendo esta una ventaja a aprovechar y no algo a evitar.

Qué es la agilidad

No existe una definición exacta de qué es agilidad. Para muchos es cumplir con los valores del manifiesto ágil. Para otros, la agilidad es una cultura, una manera diferente de ver la creación de tecnología y, en general, la creación de trabajos intelectuales. No

obstante, como afirmaba uno de los firmantes del manifiesto ágil, Robert C. Martin, la agilidad "puede ser un conjunto de valores o una cultura, pero siempre sobre la implementación de un conjunto de prácticas (reales e implantadas)"[4].

¿De qué prácticas hablamos? Pues como decía el propio Robert C. Martin, pueden ser las de Scrum o las de eXtremme Programming (populares catálogos[5] de buenas prácticas ágiles), o no, puede ser TDD[6], Integración Continua, etc. Lo normal es que sean esas, pero si se trata de otras deben ser cercanas a los valores del manifiesto ágil y sus principios. Veamos un resumen de las principales, las más características y deseadas en la actualidad.

Las relativas al equipo humano

El equipo ágil es pequeño, el tamaño óptimo no sobrepasa las 10 personas, a partir de ese número las pérdidas de tiempo y productividad por coordinación, gestión y aumento de canales de comunicación provocan demasiado desperdicio (por ello las empresas grandes ágiles dividen equipos numerosos en equipos de menos de 10 personas).

Además, el equipo ágil es auto-organizado, autónomo, adaptable (se ajusta dinámicamente según sea necesario) y responsable colectivamente de los resultados. Y es multifuncional, posee las competencias necesarias para lograr completar el trabajo sin depender (o dependiendo mínimamente) de otros equipos o departamentos.

Todo lo anterior rompe con la visión más clásica de una empresa de desarrollo, que suele ordenarse por departamentos con mucha gente, donde ninguno es capaz de completar el trabajo sin depender de otros, lo que implica 'desperdicios' y tiempos.

Ciclo de vida y el proceso

Quizá de las prácticas más características de la agilidad, hay que mencionar el uso del ciclo de vida iterativo e incremental (añadir poco a poco y frecuentemente valor al producto y dedicar explícita y regularmente tiempo a refactorizar y mejorar la calidad), con iteraciones cortas (semanas). El ciclo de vida iterativo e incremental es una de las buenas prácticas de ingeniería del software más antiguas, su primer uso en el software se data en los 50, pero solo se ha popularizado ampliamente hasta hace unos años.

Las relativas a la calidad

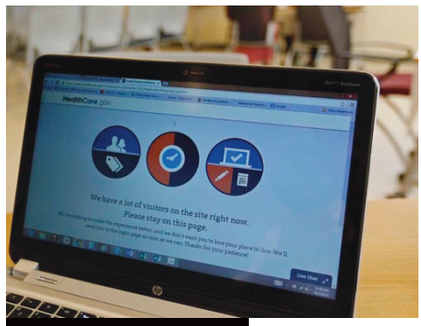
La calidad de software es importante, desde varias perspectivas y prácticas de carácter técnico: el papel clave del testing en todas sus dimensiones, la calidad del código, controlar la llamada deuda técnica (la mala calidad genera 'deudas' cuyos intereses, si se disparan, pueden llegar a ser imposibles de pagar) y el control de la versión, con técnicas como la integración continua, con el objetivo de acortar cada vez más las distancias entre desarrollo y explotación, el DevOps o continuous delivery, entre otras técnicas.

Sobrevive quien se adapta

La amplia mayoría de las prácticas, recomendaciones, postulados, ideas, etc., bajo el paraguas de la agilidad tienen más de 20 y 30 años. La agilidad, y ahí está su verdadero mérito, ha sabido rescatarlas del pasado, adaptarlas y agruparlas bajo un único nombre. Por ello, ver a la agilidad como una moda pasajera es un gran error. Desde mi primer proyecto ágil en 2001 hasta la actualidad, y después de haber podido trabajar para más de 80 empresas, la agilidad hoy es el gran reto que persiguen la mayoría de empresas de base tecnológica, disparando la competitividad de las organizaciones que lo logran.

Así, la agilidad está marcando hoy un nuevo salto evolutivo en el desarrollo software, en la tecnología en general, en los negocios de base tecnológica y en la gestión de proyectos, a todos los niveles, desde la programación a la organización y estructura de las empresas.

Un salto evolutivo como el que en su día supuso la programación estructurada, las BBDD relacionales o la OO, sobre las que en su momento se debatió mucho; hubo quien supo aprovecharse del salto y quien no supo evolucionar, y a día de hoy son saltos tecnológicos, evolución, que nadie discute. ■



La agilidad es el gran reto que persiguen las empresas de base tecnológica.

1) http://www.newyorker.com/tech/elements/dont-go-chasing-waterfalls-a-more-agile-healthcare-gov
2) http://www.nytimes.com/2013/10/25/opinion/getting-to-the-bottom-of-healthcaregovs-flop.html?_r=0
3) http://agilemanifesto.org/
4) http://blog.8thlight.com/uncle-bob/2014/03/28/The-Corruption-of-Agile.html
5) Estrictamente hablando el concepto metodología, aunque se use, no existe en agilidad, la definición más correcta serían "frameworks" de buenas prácticas
6) Test Driven Development