Software Development by Steelmood

El Desarrollo de Software por Steelmood

Fernando Ruiz-Falcó[1]

[1] Vicepresidente Steelmood

fernando.ruizfalco@steelmood.com

RESUMEN. Este artículo analiza el Desarrollo de Software en las empresas a través de la experiencia y los casos vividos por la empresa consultora, 'Steelmood'. Teniendo como guía los siguientes puntos: Lecciones aprendidas de otras industrias, La industria de fabricación de software a medida hoy, Las bases del Modelo Software Development by Steelmood, La dinámica de trabajo de un equipo en SDS, Beneficios del Modelo SDS, Barreras para implantar el Modelo SDS y El proceso de transformación y escalado.
Llegando a la conclusión de que estamos cerca de un importante cambio de paradigma en la industria del desarrollo de software a medida y presentando su propia propuesta para ayudar a las empresas a dar el salto a este nuevo paradigma cuanto antes.

ABSTRACT. This paper analyzes Software Development in companies through experience and cases experienced by the consulting firm, 'SteelMood'. Taking as a guide the following: Lessons learned from other industries, Manufacturing industry custom software today, Bases of the Model Software Development by SteelMood, Work dynamic of a team in SDS, SDS Model Benefits, Barriers to implement SDS Model and Process of transformation and scaling.
Concluding that companies are near a major paradigm shift in software development industry as and presenting their proposal to help companies make the leap to this new paradigm as soon as possible.

PALABRAS CLAVE: Desarrollo de Software, Tendencia del Desarrollo de Software, SDS, Proceso de transformación y escalado, Consultoras, Ingeniería de Software.

KEYWORDS: Software Development, Software Development Trends, SDS, Process transformation and scaling, Consulting, Software Engineering.

1. Introducción

"Every business is a software business, and every business can profit from improved software process". Watts Humphrey

Con la apertura del CAIS (Centro Avanzado de Ingeniería de Software) de Steelmood, estos últimos años, hemos tenido ocasión de hablar en profundidad con los responsables de desarrollo de software de las principales compañías españolas. Esto nos ha permitido tener una idea de los problemas reales desde el punto de vista del "gran consumidor", es decir, compañías que invierten varios cientos de millones de euros al año en desarrollar software a medida de sus necesidades. Este mercado tiene un tamaño de unos tres mil millones de euros en España y unos tres billones de dólares en el mundo.

Como síntesis de estas entrevistas, los tres principales retos a los que se enfrenta esta industria son:

- Requerimientos: cómo realizar unas especificaciones funcionales que recojan las necesidades reales del negocio y los usuarios del Sistema.
- Construcción: cómo construir el producto software en el tiempo y coste previsto y con el nivel de calidad esperado.
- Rendimiento de la máquina: cómo mejorar la gestión y el rendimiento de la máquina productiva, esto es, el retorno de la inversión realizada (ROI).

Hace treinta años, estos problemas eran los mismos, sobre todo los dos primeros. Cuando teníamos un problema de este tipo en algún proyecto, razonábamos con el cliente que ello era debido a la juventud de la industria de desarrollo y a la falta de experiencia realizando trabajos similares. Desde luego, ninguno de estos argumentos sigue siendo válido hoy: ha pasado mucho tiempo (la industria ya no es tan joven) y, además, la gran mayoría de las funciones que se están codificando hoy, ya han sido desarrolladas. Aún así, nos enfrentamos al mismo tipo de problemas porque no hemos sabido madurar suficientemente el proceso productivo como sí han hecho, de forma espectacular otras industrias, durante este periodo de tiempo.

Es increíble lo poco evolucionado que está el paradigma actual de gestión de la producción de desarrollo de software. La única explicación sencilla que podría explicar tan llamativo fenómeno es que mentes brillantes, tanto en los propios clientes como en sus proveedores, han estado tan absorbidas por el poder (y evolución) de la tecnología, que, a diferencia de lo que ha ocurrido en otras industrias, no han tenido tiempo suficiente para detenerse a reflexionar sobre el proceso productivo en sí mismo.

Repasemos brevemente lo que ha ocurrido en estas últimas décadas en las industrias que fabrican en el mundo físico (automoción, electrónica de consumo, aeroespacial, maquinaria pesada, etc.). En primer lugar, en todas ellas se ha producido una muy significativa evolución en mejora de la calidad, costes y plazos de entrega de los productos que fabrican. Pensemos, por ejemplo, en un automóvil; si comparamos su coste y calidad hace treinta años con los actuales, llegamos a la conclusión de que hoy son mucho mejores (consumo, seguridad, fiabilidad, prestaciones, etc.) y a un coste muy inferior. Este mismo razonamiento es válido para otras muchas industrias manufactureras: ordenadores, electrodomésticos, aviones, maquinaria pesada, etc.

En segundo lugar, todas estas industrias, por puro efecto darwinista, han expulsado del mercado a quienes no han sabido evolucionar (o revolucionar) sus procesos productivos para encontrar mejoras tan significativas. Por seguir con el ejemplo del automóvil, si un fabricante como Toyota rompe las reglas de la industria ofreciendo ventajas en calidad y costes a sus clientes, o los demás reaccionan, o simplemente desaparecen, ya que ¿Quién quiere comprar un coche peor o más caro?

En tercer lugar, el proceso de mejora pasa por saltos disruptivos que no son sencillos de afrontar pues implican verdaderos cambios de paradigma. Por supuesto que ni a Toyota ni a Motorola, por citar sólo dos ejemplos, les ha sido sencillo mejorar tan significativamente sus métodos productivos. Es imprescindible la visión y el com-

promiso de la alta dirección, la disciplina e insistencia en el tiempo y la involucración de todos los actores afectados (operarios, supervisores y gestores). La tarea no es fácil, pero la motivación, clara. Quien no proceda de ese modo, desaparecerá.

2. Lecciones aprendidas de otras industrias

En todos los casos que estamos citando de otras industrias, hay un hecho fundamental que ha sido condición necesaria para realizar saltos significativos de mejora: dedicar una parte importante del esfuerzo intelectual y operativo a reflexionar y mejorar el proceso productivo en sí mismo, más allá de que nos dediquemos a producir automóviles o lavadoras.

Evidentemente, otra parte del esfuerzo seguirá estando en el diseño y fabricación del propio producto, pero esto no es el todo, sino solo una parte y, en algunos casos, desde luego que no es la más significativa.

Dicho de otra forma: en una fábrica de coches trabajan ingenieros expertos en automóviles que conocen perfectamente sus componentes (motores, carrocerías, electrónica, materiales, etc.) pero también hay ingenieros que apenas saben nada de lo anterior y son expertos en los propios procesos productivos (compras, ensamblaje, LEAN, KANBAN, Six-Sigma, etc.). De hecho, cuantitativamente, en la industria del automóvil de hoy, trabajan muchos más de los segundos (ingenieros de procesos) que de los primeros (ingenieros de producto).

En realidad, la lección aprendida que obtenemos es que encontrar el balance adecuado entre ingenieros de proceso e ingenieros de producto para cada industria en cada momento de mercado, es clave para el éxito. Es decir, es esencial acertar con las dosis necesarias de esfuerzo en producto y esfuerzo en proceso necesarias para satisfacer las expectativas del cliente.

La segunda lección es que es imprescindible la visión, determinación y compromiso de la alta dirección en conseguir mejoras importantes y, además, disponer de un modelo estructurado para implantarlas. Sin la concurrencia de ambas, únicamente es posible tener éxito por casualidad. Y cuanto menos cosas relativas a nuestra propia subsistencia dejemos en manos de la diosa Fortuna, mejor.

La visión, determinación y compromiso es imprescindible como en cualquier caso que implique un cambio de paradigma, esto es, que se proponga una nueva utopía más allá de lo razonable. Si John Fitzgerald Kennedy no hubiera anunciado en mayo de 1.961 su visión de que un americano pisaría la Luna antes de terminar la década y no hubiera puesto toda su determinación para hacerlo, Neil Amstrong probablemente no podría haber dicho su famosa frase "un pequeño paso para el hombre pero un gran salto para la humanidad" al pisar el suelo de nuestro satélite el 21 de julio de 1.969. ¡Únicamente ocho años después!

Pero es también necesario que esta energía sea canalizada por un modelo que facilite su puesta en práctica de forma estructurada, sistemática, eficaz y eficiente.

La tercera y última lección es que el gran salto se inicia poniendo unas pocas personas a pensar en profundidad sobre la ingeniería del proceso productivo, pero el impulso fuerte realmente se produce cuando la mentalidad y actividades concretas relativas a la mejora de la calidad de los procesos se extienden por todas las personas que participan en el proceso productivo.

Veamos como resumen esta evolución los profesores Jay Rao y Fran Chuán:

"En la posguerra de la Segunda Guerra Mundial, los profesores Demin y Juran, enseñaron técnicas estadísticas y de gestión de la calidad a empresas japonesas. El centro de atención en ese momento era la inspección, la reacción a los defectos y el control de calidad. En los años 70 la responsabilidad por la calidad era mucho mas funcional y estaba limitada a unos pocos trabajadores de la empresa (…) En 1980 la norteameri-

cana Naval Air Systems Command introdujo el término TQM (Total Quality Management). Este nuevo concepto incluía la prevención de defectos y hacía a todos los trabajadores responsables de la calidad. El hecho de que fueran muchos los proveedores que trabajaban con Naval Air Systems Command hizo que muchas compañías empezaran a conocer y adoptar la gestión de la calidad total, y que esta se propagara como el fuego (…) El SIx-Sigma ha sido la última evolución en el movimiento de la calidad. Aunque sigue manteniendo algunos principios estadísticos y procesos fundamentales de los métodos anteriores, se trata de un método más deliverado, sistemático y exhaustivo que aquellos, lo cual representa un gran paso hacia delante. Desarrollado a mediados de los años ochenta por Motorola, Six-Sigma salta a la fama cuando Motorola gana en 1988 el premio nacional a la calidad Malcolm Baldrige, siendo la primera vez que se presentaba." (2012: 16)

3. La industria de fabricación de software a medida hoy

Es evidente que la industria de desarrollo de software a medida está muy atrasada en el proceso evolutivo que acabamos de describir. Todavía no ha terminado de dar el primer paso para incorporar el control estadístico de los procesos y filtros de calidad que propuso Deming y, mucho menos, para involucrar a todas las personas en el compromiso de mejorar en costes y calidad.

Hoy, el estado del arte de la industria, se limita a conformarse con adoptar etiquetas formales (ISO, CMMI, etc.) más centradas en el qué hay que hacer para obtener una certificación que cómo se mejora realmente, sin abordar los problemas reales desde los propios equipos de trabajo.

Es decir, se trata, en el mejor de los casos, de pequeños grupos dedicados a la mejora de procesos luchando contra todos los elementos (clientes, equipos de trabajo, etc.), donde es escaso el compromiso de la alta dirección y el de los propios equipos de trabajo. Los modelos de trabajo aportan mejoras parciales pero no son en absoluto una aproximación holística que involucre a toda la organización de modo sistemático.

De esta manera, más del 90% del esfuerzo de los equipos de trabajo se centran en la ingeniería de producto, cubriendo su dimensión tecnológica y funcional, pero infraponderando estrepitosamente la dimensión de mejorar el proceso productivo en sí mismo.

Una imagen sencilla de la situación general es que los equipos tienen tanta leña que cortar, que no creen tener tiempo para afilar el hacha, cuando, visto desde fuera, es evidente lo rentable de la inversión en el afilado.

Este escaso avance en lo esencial, no significa que durante estos años, tanto clientes como proveedores, no hayan realizado ninguna evolución. Es conveniente señalar tres significativos:

Tecnología. Es evidente el avance que se ha producido en este campo. Evolución en lenguajes de programación, hardware, sistemas operativos, redes de comunicación, dispositivos, gestores de bases de datos, etc.

De hecho, asumir el esfuerzo de evolución en este campo, es la razón fundamental de la poca evolución en el afilado del hacha: las nuevas tecnologías eran tan espectaculares que se convertían en el objetivo mismo, perdiendo la perspectiva de que, en general, la tecnología es un medio al servicio del negocio más que un objetivo en sí misma.

Modelos. Aunque les falte por alcanzar un uso masivo y no sean completos para abordar todas las componentes de forma integrada, sí se han producido avances en algunos campos, generalmente por el lado académico. A continuación, se citan los más significativos que hemos introducido como componentes en nuestro modelo de desarrollo SDS: TOGAF para Arquitectura Empresarial, BABOK (del International Institute for Business Analyst) para gestión de requerimientos, Personal Software Process (PSP) y Team Software Process (TSP), del Software Engineering Institute y Carnagie Mellon University, para la construcción, conceptos TSP

y AGILE para la organización de equipos y LEAN (Lean Advancement Initiative, Massachusetts Institute of Technology) para el análisis y diseño de procesos.

Maduración del proceso de compra-venta. La presión competitiva del mercado, al carecer de un modelo que permita gestionar el coste total del desarrollo de software, se ha centrado en la disminución del coste unitario, es decir, el precio por hora de la mano de obra interviniente. De esta manera se han firmado estos años muchos acuerdos marco o contratos de externalización de desarrollo y mantenimiento con fuerte presión en el coste horario de la mano de obra como principal parámetro de negociación (tanto con proveedores locales como globales). Esta situación fue bien aprovechada por compañías indias, ofreciendo mano de obra bien cualificada a coste horario sensiblemente inferior.

Consecuentemente, este recorrido ha llegado al extremo del péndulo el cual está a punto de dar la vuelta y empezar a moverse en dirección contraria. Los clientes se han dado cuenta de que han conseguido disminuir sensiblemente el coste por día trabajado, pero, al mismo tiempo, ha aumentado el coste total, que es su problema real. Y, además, en los casos de negociación más radical en precio, la calidad ha empeorado. Más adelante, analizaremos en mayor profundidad las razones y vías de solución de esta paradoja, pues entendemos que va a ser (o está siendo ya) una de las fuerzas principales que impulsarán el cambio de paradigma en la industria de desarrollo de software a medida.

Por todo ello, es un hecho ya en el mercado que grandes compradores de Off-Shore en países como Reino Unido y Holanda, por ejemplo, y que firmaron grandes contratos de externalización con compañías basadas en India, no los están renovando y están buscando otras soluciones (In-site o Near Shore). En España, aunque la proporción de mercado con India es menor, el fenómeno es el mismo, con proveedores locales o globales, con fuerte presión en coste unitario.

En resumen, por su propia experiencia y a base de disgustos, los clientes hoy ya han aprendido que el coste unitario de la mano de obra es uno de los parámetros a tener en cuenta a la hora de contratar con un proveedor, pero no el único si queremos mejorar efectivamente el problema real para el cliente que no es otro que mejorar la rentabilidad de su inversión en software.

Para poder abordar el problema real es necesario introducir otras dos dimensiones nuevas a la ecuación: la productividad y la calidad del producto entregado. Ilustremos esto con un ejemplo.

En efecto, es más barato pagar, digamos, 200 € por día por una persona (o proveedor) (A) con una productividad de 30 líneas de código por hora que 100 € por otra (B) con una productividad de 10 líneas de código por hora. En el primer caso la línea de código cuesta 0,83 € y en el segundo 1,25 €. El coste de producir una línea de código (o un punto función) no refleja completamente el coste total, pero ya es una aproximación mucho más cercana a la realidad que considerar únicamente el coste horario.

En este ejemplo tan sencillo (y tan frecuente en estos últimos años) vemos como una disminución del 50% en coste unitario puede producir un resultado de un aumento del 50% del coste total.

Para incorporar al ejemplo la calidad debemos comenzar por realizar una breve reflexión sobre el asunto, comenzando desde la raíz.

Dado que en el proceso intervenimos humanos, vamos a cometer errores e insertarlos al producto. Aquí el software se comporta como cualquier otro proceso productivo y nos sirve todo lo aprendido en otras industrias (TQM, Six-Sigma, etc): cuanto antes se corrija el error en la cadena productiva, más barato será hacerlo.

En el caso del software, en palabras de Manies y Nikual:

"Es una realidad que entre el 40% y el 60% de los errores y defectos del software son el resultado de una

pobre gestión y definición de requisitos. En palabras simples, esto significa que aproximadamente la mitad de los problemas encontrados se podrían haber evitado, simplemente, con dejar claro desde el principio, lo que el cliente espera del proyecto."(2011: 26)

La mayoría de las veces los errores en requerimientos no se detectan hasta la fase de pruebas, lo que implica enormes decepciones, retrasos y sobre coste por retrabajo.

En este punto conviene introducir el "Failure Appraisal Ratio" pues es el indicador que nos ayuda a balancear el esfuerzo al detectar errores en etapas tempranas optimizando el resultado final. Se define como la división entre el tiempo total dedicado a la prevención de errores (filtros de calidad) y el tiempo total dedicado a pruebas. De acuerdo con la literatura actual, el valor óptimo para software de gestión está alrededor de 2, es decir, cuando dedicamos el doble del tiempo en filtros intermedios que en pruebas.

¿Cuánto es este ratio en su organización? Si es algún valor entre cero y uno, como es habitual hoy, tiene usted ante sí una enorme oportunidad de mejora.

Ahora estamos en disposición de incorporar el asunto de la calidad a nuestro pequeño ejemplo. La pregunta fundamental es ¿Cuánto cuesta que se produzca un error después de la liberación del producto? Este coste tiene dos componentes:

- El coste económico y en disminución de la satisfacción del cliente, derivado de la suspensión del servicio interrumpido por el fallo de la aplicación. Puede ser enorme pero no es sencillo de estimar consensuadamente.
- El coste de los técnicos que analizan y reparan el error. Aunque puede ser mucho menos significativo que el anterior, es fácil de calcular y es el que suele incorporarse a los modelos como beneficio cuantificable (y dejando el anterior como no cuantificable).

Evidentemente, el tiempo que se tarda en arreglar un error, una vez liberado el producto, es un parámetro sensible a la aplicación e instalación en concreto. Si usted dispone de información de su propio caso, utilícela; pero mientras dispone de su propia serie histórica, puede considerar las 20 horas que calcula Standish Group como el tiempo medio necesario para corregir un defecto después de la liberación del producto.

Por lo tanto, para completar en el ejemplo el coste total de cada caso para el cliente, necesitamos incorporar la tasa de defectos que incorpora cada uno. Supongamos que el primero tiene un coste de 200 € por día, una productividad de 30 líneas de código por hora e incorpora un defecto por cada mil líneas de código y que el segundo cuesta 100 € por jornada, produce 10 líneas de código por hora e incorpora una tasa de 10 defectos por cada mil líneas de código.

Según lo visto hasta ahora, en el primer caso el coste de producir mil líneas de código es 0,83*1000 = 830 € y en el segundo 1,25*1000 = 1.250 €. Ahora podemos saber además, lo que nos costará mantener el producto en cada caso.

En el primer caso tendremos un defecto que nos costará 20 horas (2,5 jornadas) arreglarlo, es decir 2,5 jornadas * 200 € = 500 €. El coste total del sistema en este caso es 830 para su desarrollo y 500 para su mantenimiento, es decir 1.350 €.

En el segundo caso, hemos de esperar 10 defectos que nos costará arreglar 20 horas cada uno, es decir 10* 2,5 jornadas * 100 € = 2.500 €. El coste total del sistema en este caso es 1.250 € para su desarrollo y 2.500 € para su mantenimiento, es decir 3.750 €.

En resumen, en el ejemplo vemos un caso en el que se disminuye el coste de la mano de obra a la mitad y aumenta un 177% el coste total y aunque parezca disparatado, no lo es tanto como algunos ejemplos reales

de grandes compañías estos últimos años.

De este análisis de la situación actual de la industria del desarrollo de software a medida, conviene extraer dos conclusiones importantes. En primer lugar, para empezar a gestionar la totalidad del asunto, los clientes necesitan incorporar parámetros de productividad y calidad en sus procesos de compra y de gestión. Y la segunda es que este hecho lo han descubierto por sí mismos, o están cerca de hacerlo, basados en su propia experiencia de estos últimos años. Los proveedores, en general, estábamos demasiado ocupados incorporando nuevas tecnologías y disminuyendo nuestros costes unitarios de la mano de obra.

Esta insatisfacción con el modelo actual acumula considerables dosis de energía para impulsar el cambio, cuando aparezca la palanca adecuada para canalizarla.

4. Las bases del Modelo Software Development by Steelmood

Hemos llamado al modelo Software Development by Steelmood para señalar que, aunque el modelo es característicamente nuestro, lo hemos construido utilizando como componentes otros modelos y estándares internacionales en diversos campos, como los citados anteriormente. Hemos trasladado al desarrollo de software de gestión a medida, alguna de las ideas básicas que se han producido en la ingeniería de procesos de las industrias manufactureras anteriormente analizadas.

Veamos primero sus principales características y después sus bases conceptuales.

La primera característica de SDS es que es un modelo disruptivo, esto es, su puesta en marcha implica un cambio de paradigma en la organización del cliente. Este cambio será mayor o menor según el nivel de madurez del cliente, pero es siempre significativo pues implica un cambio profundo en la forma de realizar las actividades por los propios equipos de trabajo de desarrollo.

La segunda característica de SDS es que es un modelo holístico. Su objetivo es abordar todos los asuntos relacionados con el desarrollo de software a medida, con aportaciones de diversas disciplinas para poder abarcar de forma integrada todos los aspectos del asunto.

La tercera característica de SDS es que es un modelo pragmático. Su finalidad es la mejora de los resultados, es decir, del retorno de la inversión de nuestros clientes en desarrollo de software a medida.

La primera base conceptual del modelo es introducir procesos sistemáticos y cuantitativos de mejora en tres capas:

- La persona individual. Se trata de ayudar a cada ingeniero de software a que mejore su propio desempeño personal. Para ello, nos basamos en las ideas extraídas de PSP (Personal Software Process es Service Mark de la Universidad de Carnegie Mellon), del coaching ejecutivo y de la gestión del cambio personal de Goullart y Kelly.
- El equipo de trabajo. El propósito es mejorar el desempeño de cada conjunto de ingenieros de software trabajando en un proyecto común. Para ello nos basamos en las ideas extraídas de TSP (Team Software Process Process es Service Mark de la Universidad de Carnegie Mellon), del coaching ejecutivo y de la gestión del cambio en un equipo natural de trabajo de Goullart y Kelly. Y la gestión de la calidad percibida de Noriaki Kano.
- Procesos organizacionales. Para promover la esbeltez del flujo de procesos corporativos del cliente relacionados con el desarrollo de software, con los que, de una u otra forma, han de interactuar (y respetar) los equipos de trabajo de desarrollo. Nos basamos en ideas extraídas de LEAN (promovida por el LAI, Lean Advancement Initiative, MIT) y las 4R´s del cambio y Arquitectura de la Movilización de Goullart y Kelly.

El eje central de la mejora del desempeño individual y colectivo es enseñar a registrar con granularidad y rigor la información sobre el tiempo invertido en construir cada artefacto, el tamaño del mismo y los defectos que insertamos. Esto significa:

Tiempo: Se trata de registrar el tiempo realmente invertido en desarrollar el artefacto. La suma de estos tiempos no es ocho horas al día, ni cuarenta horas a la semana. No se trata de repartir el tiempo que trabajamos entre distintos criterios de actividad o imputación, que es lo que habitualmente se hace en la actualidad. La lógica es distinta y el objetivo también y es clave explicar esto bien, en los cursos de inmersión, a los ingenieros neófitos en el modelo, para que puedan aplicarlo correctamente. No se trata de justificar o repartir; se trata, simplemente, de registrar el tiempo verdaderamente invertido en cada artefacto, cualquiera que éste sea. La recogida de esta información se puede realizar de forma automática en la propia plataforma de trabajo.

Tamaño: Se registra el tamaño de los artefactos producidos. Evidentemente, es necesario definir una métrica de tamaño para cada tipo de producto. Nuestro modelo es agnóstico respecto a la elección de la métrica concreta, no vemos una ventaja significativa por utilizar una en vez de otra, sino que, lo único verdaderamente importante, es ser coherente con la métrica elegida y utilizar siempre la misma para poder comparar. Así, por ejemplo, para medir el tamaño de un programa, nos parece válido utilizar número de líneas de código, puntos función o casos de uso; para análisis funcional, número de páginas, líneas o palabras, etc.

Defectos: Para los defectos, es necesario definir una taxonomía de errores para cada tipo de artefacto: análisis funcional, diseño técnico, codificación, pruebas, etc. Para ello es importante adaptar adecuadamente el nivel de sofisticación de la clasificación propuesta en los manuales del modelo con el nivel de madurez de los equipos involucrados. Se definen en el grado de detalle adecuado para la mejora de los resultados, aconsejando huir de definiciones excesivamente complejas en un principio, para ir evolucionando en la lista con su uso.

Granularidad y rigor estadístico: Los registros tienen una granularidad de horas, minutos y segundos. Algunos productos los desarrollamos en algunos segundos y, para otros, invertimos muchas horas. En ambos casos, lo mediremos con exactitud para que los datos históricos correlacionen estadísticamente. Así nos aseguramos que no estamos introduciendo ningún sesgo con el procedimiento de medida. Esto es fundamental por varias razones:

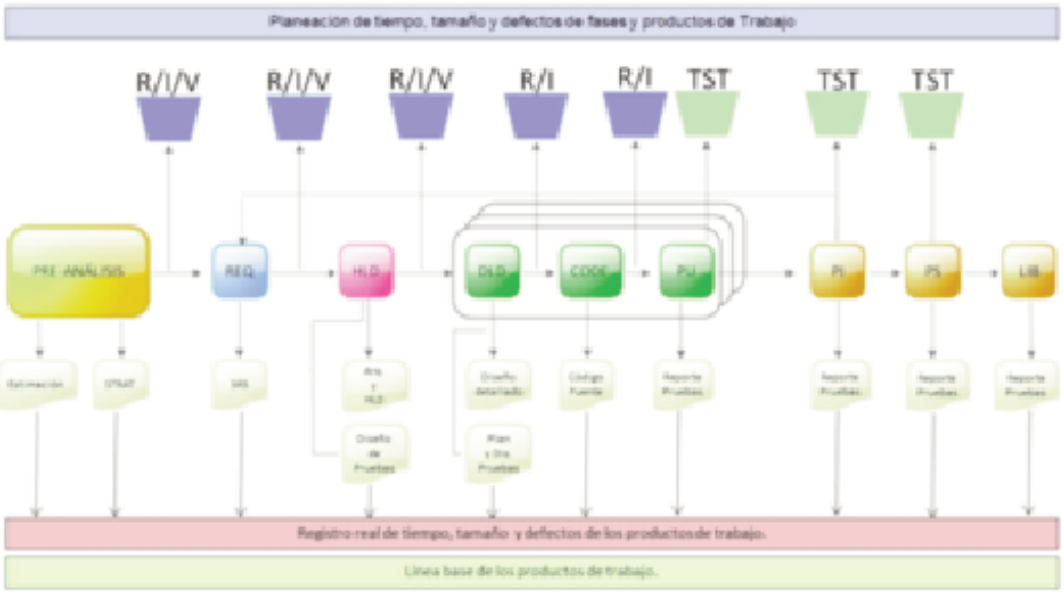
La primera es que si comenzamos a tener datos estadísticos sobre nuestro comportamiento, empieza a ser posible predecir el mismo con márgenes de errores estadísticos. Esto supone un enorme avance en términos de predictibilidad de las entregas y, en consecuencia, de time-to-market. También es clave para la estimación de carga de nuevos proyectos, SLA´s de productividad y calidad, a los que realmente nos podemos comprometer.

Y en segundo lugar, porque tenemos información de partida para construir un conjunto completo de métricas que facilitan la mejora de los individuos, la mejora de los equipos y las decisiones de los supervisores y directivos.

En la siguiente ilustración, proporcionamos la lista completa de métricas:

El segundo punto conceptual básico es establecer filtros de calidad inmediatamente después de cada paso del proceso productivo hasta equilibrar el Appaissal Failure Ratio. Con estos filtros de proceso evitamos arrastrar errores a lo largo del ciclo de vida. Cuanto antes removamos el error insertado, más barato será hacerlo, evitaremos retrabajos y mejoraremos la calidad del producto resultante.

En el diagrama siguiente, mostramos el mapa general de los pasos y artefactos en el ciclo de vida de un desarrollo de software. Obsérvese que, en cada paso del proceso, hay un filtro con los códigos R, I y V. La R significa realizar una revisión (el propio autor del producto revisa el mismo para corregir errores), I simboliza una inspección (otra persona distinta al autor, con similar cualificación profesional, inspecciona el producto para corregir errores). Y la V significa validación, donde es algún representante del cliente el que valida el producto, señalando sus posibles errores.



En este punto, conviene ver algún ejemplo. Por su serie histórica, sabemos que el Analista A escribe análisis funcionales con una productividad de 0,77 páginas por hora, insertando un defecto mayor y 20 menores cada 100 páginas. Además, el 80% de los defectos mayores que ha insertado históricamente el Sr. A son de completitud del requerimiento, el 10% son de consistencia y el otro 10% se reparte entre factibilidad y agregación; y el 60% de los defectos menores son de ortografía, el 20% de gramática y el otro 20% de contenido menor. Además, identifica la mitad de los defectos que inserta realizando inspecciones de su propio trabajo, con una productividad de revisión de 10 páginas por hora.

Naturalmente, el primero que dispone de toda esta información es el propio Sr. A, y con un poco de ayuda de un coach, le será muy útil como instrumento básico para la mejora de su propio trabajo. No olvidemos el principio básico de que podemos mejorar lo que somos capaces de medir.

Pero la información también está disponible para los demás miembros del equipo, de manera que si el Sr A reporta que ha escrito un funcional de 100 páginas en 20 jornadas de trabajo y que lo ha revisado durante 4 horas y ha corregido cero defectos mayores y cinco menores, el supervisor sabe inmediatamente que:

La productividad del Sr A en este trabajo (0,65 páginas por hora) ha sido algo inferior a su productividad media (O,77 páginas por hora). O este funcional es un poco más complejo de lo normal, o la motivación del Sr A está más baja de lo normal, o ha habido algún problema o complicación especial.

Sin embargo, la velocidad de revisión en este caso (25 páginas por hora), es 2,5 veces superior a su media (10 páginas por hora). Esto significa que tengo que esperar que aún queden defectos sin identificar en el funcional y puedo decidir que se realice otra inspección o revisión.

El tercer punto conceptual básico es disponer de una herramienta que facilite la imputación de datos y calcule y ponga accesibles, en sus diversos grados de agregación, todas las métricas.

Con la información de estas métricas y un poco de adiestramiento (un curso de una semana intensiva y un periodo de un año de coaching en el equipo de trabajo), se comienzan a tomar decisiones basadas en datos objetivos en los tres niveles mencionados anteriormente: individual, equipo de trabajo y organización en su conjunto.

Y el cuarto punto básico conceptual es medir también la calidad subjetiva, como parámetro básico de la relación contractual. Para ello, como hemos señalado anteriormente, utilizamos las ideas de calidad subjetiva de Noriaki Kano, incorporadas al procedimiento EAP (Expectations Aligment Process) definido por Steelmood:

Al comienzo del proyecto o servicio pedimos al cliente que nos explique sus expectativas o parámetros de satisfacción subjetiva para este proyecto o servicio. Así comprendemos los criterios de la calidad percibida definidos por el cliente para cada caso, pudiendo ponderar cada uno según importancia de cada criterio (de 0 a 10).

Al final del mismo, o en los periodos de revisión pactados, el cliente evalúa de 0 a 10 su satisfacción obtenida en cada uno de los criterios, explicando las razones de su puntuación, proporcionando feedback.

Si el resultado es inferior a 7, nos aplicamos una penalización y devolvemos al cliente el 20% del importe facturado.

El objetivo fundamental de este procedimiento es de alineamiento con el cliente. De esta manera, nos aseguramos que estamos haciendo lo que realmente es importante para él, aunque sea difícil de medir.

5. La dinámica de trabajo de un equipo en SDS

Acabamos de resumir las principales características y bases conceptuales del Modelo SDS. A continuación, describimos la dinámica de trabajo de un equipo SDS, que es la célula fundamental del Modelo.

Son equipos autodirigidos, que reparten entre sus miembros todos los roles administrativos y de control de procesos, y realizan lanzamientos, ciclos y postmortem, típicamente en iteraciones de unas seis. Obsérvese que todos estos conceptos son análogos y muy compatibles con metodologías AGILE tipo SCRUM, pero siempre con la consideración de que, en el Modelo SDS, las decisiones se toman basadas en métricas cuantitativas.

En resumen, SDS asegura que, de forma sistemática, los principales actores del aparato productivo (ingenieros de software y equipos de desarrollo) dediquen parte de su esfuerzo a reflexionar y mejorar su propio proceso productivo, facilitando a supervisores y directivos la toma de decisión más adecuada, basada en datos objetivos y medibles.

6. Beneficios del Modelo SDS

Si clasificamos los beneficios por su carácter financiero o no financiero (operativo), y, , a su vez, cuantificables o no cuantificables, el Modelo proporciona beneficios en los cuatro cuadrantes.

A continuación se facilita una lista de los principales beneficios generales que se pueden particularizar en un "Business Case" para cada caso, de acuerdo con sus propios objetivos y circunstancias:

- Ahorro significativo en el coste total del desarrollo (entre el 25% y el 50%, según el caso concreto). Este ahorro se basa en:
  - Disminución drástica (alrededor del 80%) del coste del mantenimiento correctivo derivado de una mayor calidad del producto, en el momento de su liberación (alrededor de 0,4 defectos por cada mil líneas de código).
  - Reducción a la mitad del tiempo dedicado a pruebas debido a la supresión de errores en los filtros previos.
  - Reducción del coste del desarrollo propiamente dicho cercana al 20% por disminución de retrabajos y vueltas atrás en el proceso.
- Ahorro en suspensiones de servicios de negocio por disminución de errores en producción, deri-

vados de la mayor calidad del producto liberado.

- Mejor time-to-market por mejora de predictibilidad de las fecha de entrega.
- Mayor satisfacción de los usuarios por la mejor calidad de los productos producidos y mayor cumplimiento de compromisos de fechas de entrega.
- Mayor motivación y compromiso de los equipos de trabajo.

7. Barreras para implantar el Modelo SDS

Las principales barreras que nos hemos encontrado para implantar el Modelo son tres:

1. Decisión y compromiso del cliente con la mejora. Es la barrera más fuerte y más difícil de tratar y, como hemos señalado anteriormente, es un ingrediente absolutamente imprescindible

2. Resistencia al cambio de las personas y los equipos de trabajo. Al final, el Modelo altera la esencia misma de su trabajo al introducir rutinas y disciplinas nuevas. El tratamiento de esta barrera es mucho más sencillo de lo que aparenta y se basa en la transparencia, la información y la gestión de un cambio de paradigma. En los cursos de capacitación en el modelo, se enseña a los participantes que los primeros beneficiarios de la aplicación de modelo son ellos mismos, pues mejorará su calidad de vida y su desempeño profesional.

3. Aparente dificultad para escalar rápidamente el Modelo a grandes volúmenes (miles de personas). Al igual que la anterior, es más su apariencia que la realidad. Como desarrollamos en el epígrafe siguiente, es posible industrializar el proceso de transformación y escalar rápidamente su implantación en varias fases con alcances y riesgos acotados.

8. El proceso de transformación y escalado

La esencia del proceso de transformación consiste en que, al menos una célula de desarrollo (equipo de 3 a 20 personas), comience a trabajar con el método simultáneamente. Esto se puede conseguir de dos maneras distintas: capacitando en el Modelo SDS a un equipo de trabajo (propio o de otro proveedor) ó externalizando el trabajo al CAIS.

Ambos procedimientos tienen ventajas e inconvenientes que aconsejarán uno u otro camino para cada caso. Lo verdaderamente importante es producir la personalización del Modelo SDS para el caso propio (protocolo de actuación, taxonomía de defectos, etc.) y comenzar a medir nuestra propia serie histórica.

Esto es, personalizar y arrancar el modelo con una simple célula de trabajo que pueda servir como grupo de referencia (benchmark de métricas y procesos).

A partir de este punto, se pueden añadir células de trabajo (o grupos de células de trabajo) a gran velocidad por uno u otro camino (externalización o capacitación de equipos existentes). En todo caso, es fundamental industrializar el proceso de capacitación debidamente personalizado por segmento (ingeniero, supervisor y directivo), mantener un equipo de referencia custodio de la evolución del modelo y punta de lanza del mismo y diseñar una adecuada arquitectura de movilización para todas las partes afectadas de la organización.

Por tanto, es perfectamente factible escalar el uso del modelo de forma rápida y controlada hasta alcanzar grandes volúmenes.

Los resultados finales mejoran con el tiempo en un doble sentido. En primer lugar, porque una parte importante del beneficio se produce en la fase de mantenimiento y el ahorro será más apreciable en el resultado final según aumente la proporción de producto desarrollado con SDS frente al total del producto en explotación.

Y en segundo lugar, por el efecto de la curva de aprendizaje, los profesionales no alcanzan todo su potencial hasta dos o tres años de experiencia real en el uso del modelo.

Sin embargo, se apreciarán claramente sus primeros beneficios desde el primer proyecto ejecutado de esta manera. Normalmente a los 2 o 3 meses de uso se empiezan a apreciar mejoras en el desempeño de los equipos.

9. Conclusiones

Estamos cerca de un importante cambio de paradigma en la industria de desarrollo de software a medida. Este cambio será similar en magnitud y resultados a los que han ido sufriendo diversas industrias manufactureras durante estas últimas décadas, de las que se pueden extraer lecciones aprendidas válidas.

Con esta visión en la mente, en Steelmood hemos desarrollado un modelo propio (SDS) para ayudar a nuestros clientes a dar el salto al nuevo paradigma cuanto antes. Desde luego que no será fácil conseguirlo, pero es imprescindible intentarlo si uno quiere subsistir. Aunque, como le gustaba decir al propio Deming, "al fin y al cabo, subsistir no es obligatorio".

Cómo citar este artículo / How to cite this paper

Apellido, A., Apellido, F., y Apellido, M. (2014). Software Development by Steelmood. International Journal of Information Systems and Software Engineering for Big Companies (IJISEBC), Vol. 1, Num. 1, pp. 38-50. Consultado el [dd/mm/aaaa] en www.ijisebc.com

References

Gouillart, Francis J. y Kelly, James N. 1.996. Transforming the Organization. McGraw-Hill Inc., 1.996.

Humphrey , Watts S. 2000. The Personal Software ProcessSM (PSPSM). TECHNICAL REPORT. TECHNICAL REPORT CMU/SEI-2000-TR-022 ESC-TR-2000-022. Team Software Process Initiative. Carnegie Mellon Software Engineering Institute November 2000.

Humphrey , Watts S. 2000. The Team Software ProcessSM (TSPSM). TECHNICAL REPORT. CMU/SEI-2000-TR-023 ESC-TR-2000-023. Team Software Process Initiative. Carnegie Mellon Software Engineering Institute November 2000.

Manies, M. & U. Nikual, U. 2011. "La Elicitación de Requisitos en el contexto de un proyecto software". Ing. USB Med, Vol. 2, No. 2, pp. 25-29. ISSN: 2027-5846. Jul-Dic, 2011. Rao, Jay y Chuán, Fran. 2012. Innovación 2.0 ¿Por qué cuando hablamos de innovación nos olvidamos de las personas? Editorial Profit.

Lean Advancement Initiative http://ssrc.mit.edu/programs/lean-advancement-initiative-lai

El modelo de Kano y gestión de expectativas http://steelmood.blogspot.com.es/2014/03/como-gestionar-las-expectativas-de-los.html y http://www.isixsigma.com/tools-templates/kano-analysis/