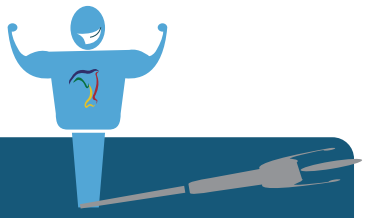
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

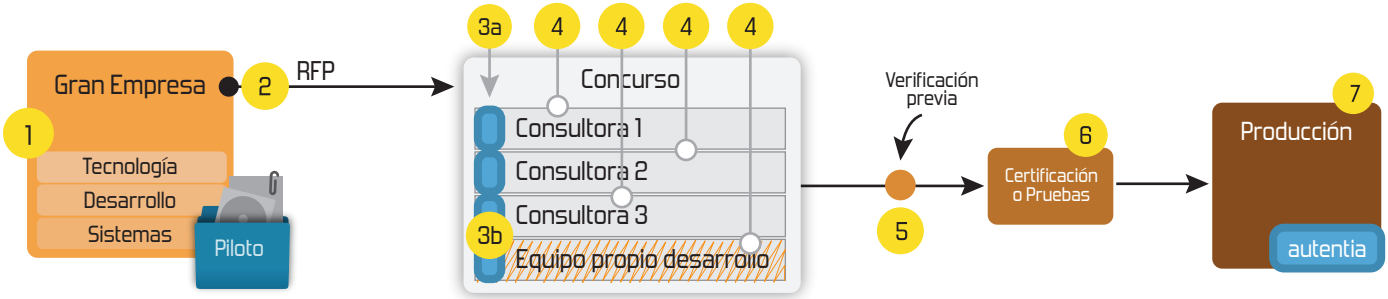
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

DESARROLLADO POR:
Alejandro Pérez García

Alejandro es socio fundador de Autentia y nuestro experto en J2EE, Linux y optimización de aplicaciones empresariales.

Ingeniero en Informática y Certified ScrumMaster

Si te gusta lo que ves, puedes contratarle para darte ayudate con soporte experto, impartir **cursos presenciales** en tu empresa o para que **realicemos tus proyectos como factoría** (Madrid). Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Fecha de publicación del tutorial: 2010-09-03 183

ï»¿

Reunión Madrid Ágil 01-09-2010: Contratos ágiles y Cómo vender TDD

Creación: 02-09-2010

Índice de contenidos

1. Introducción
2. Contratos ágiles
3. Cómo vender TDD

4. Conclusiones
5. Sobre el autor

Histórico de NOTICIAS

1. Introducción

El 01-09-2010 tuvo lugar una de las reuniones de Madrid Ágil (para mi la primera, y espero que no la última), en la que se habló de Contratos ágiles y de Cómo vender TDD.

En este artículo voy a contar mis impresiones después de la reunión.

2. Contratos ágiles

Todos los contratos se pueden englobar en dos grupos:

- **Contratos por servicio**: donde el coste es variable en función de del tiempo (da igual que el pago sea por hora/hombre, o que sea al final de cada sprint sin tener un número de sprints fijo, ...). La idea es que el cliente irá pagando según se avanza en el proyecto, sin existir una limitación prefijada de coste o tiempo.
- **Contratos por obra**: son aquellos donde el coste es fijo desde el principio. El tiempo no tiene porque ser fijo, de hecho distintos proveedores con el mismo coste y alcance podrían terminar el proyecto en fechas distintas.

En los **contratos por servicio** no hay mucho problema para adaptarlos a un contrato ágil, ya que se irá haciendo el trabajo, se irán haciendo entregas periódicas y tempranas, y el cliente irá pagando según se avanza, y podrá parar cuando quiera (bien sea porque se han cubierto sus necesidades o porque no le satisface la experiencia con ese proveedor y quiere cambiar).

Esta claro que el contrato por servicio es la mejor opción para trabajar con metodologías ágiles, pero este tipo de contrato no suele ser el más habitual, sobre todo en las primeras contrataciones, cuando todavía el proveedor no se ha ganado la confianza del cliente. En estos casos el cliente suele exigir un contrato a coste fijo que le permita controlar su inversión (nunca se va a gastar más de lo estipulado) y así evaluar las capacidades del proveedor.

En **contrato por obra o coste fijo** ¿cómo podemos encajar las metodologías ágiles? Bueno, una buena opción es hacerlo como se ha hecho siempre, es decir: se toman los requisitos, se evalúan, se priorizan se estiman, y se hace un release plan de, con ese coste fijo, cuantos de de esos requisitos se pueden llegar a completar, y cuanto tiempo se tardaría.

Pero entonces **¿donde está la agilidad?** En estos casos las metodologías ágiles pueden servir como valor añadido en el cliente, es decir, por contrato podemos acordar con el cliente que se le harán entregas periódicas para que pueda revisar el correcto avance del proyecto, y lo que es más importante, le vamos a dejar cambiar los requisitos del proyecto.

Si el coste es fijo **¿cómo le vamos a dejar cambiar los requisitos del proyecto?** ¿estamos locos o qué? Pues no, no estamos locos, la clave está en la estimación inicial de los requisitos. En esta estimación se determina cuanto "pesa" cada requisito, de forma que el proyecto es como un barco, los requisitos son la carga que lleva el barco, y el coste fijo es la línea de flotación. Si durante el desarrollo del proyecto el cliente quiere añadir un requisito, este se tendrá que estimar para ver cuanto "pesa", y habrá que negociar con el cliente que otros requisitos de similar "peso" se quitan del barco. Así mantendremos la linea de flotación siempre en el mismo nivel (coste fijo) y el barco no se hundirá. Igualmente, si el cliente quita un requisito, sabrá que puede añadir nuevos requisitos por un "peso" equivalente.

Últimos Tutoriales

SQL Joins explicados de forma gráfica con diagramas Venn

Introducción básica a la herramienta DBSchema

MediaWiki - NamespacePermissions

Prey, localizador de dispositivos móviles robados

Edición de vídeo (III): Añadir un logo de fondo transparente

Últimos Tutoriales del Autor

Ejemplo de arquitectura propuesta por Autentia

EGit un plugin de Eclipse para el sistema de control de versiones distribuido Git

Git y cómo trabajar con un repositorio de código distribuido

Gestión de proyectos ágiles con Pivotal Tracker

Eclipse Helios, la nueva versión 3.6 de Eclipse

Dentro de este marco se puede ofrecer al cliente todas las ventajas de las metodologías ágiles (entregas tempranas, control del avance del proyecto, cambios en los requisitos, ...) a un coste fijo.

Eso si, los términos del contrato han de estar muy claros para que el cliente entienda este "juego" de meter y sacar cajas del barco, y se ha de hacer un control estricto de los cambios que se van pidiendo (estricto no quiere decir pesado ni burocrático, simplemente que hay que apuntar los cambios, por ejemplo en una herramienta estilo Bugzilla o Pivotal Tracker); de lo contrario se podría acabar en una situación de "barra libre" (el cliente sólo añade cajas al barco, pero nunca quita), situación que no le interesa ni a clientes ni proveedores.

Todo esto es muy bonito pero **¿qué pasa si nos equivocamos en nuestra estimación inicial de requisitos?** Bueno en este caso simplemente estaremos en la misma situación que si hubiéramos hecho un contrato por obra tradicional, perderemos dinero. En lo que si nos puede ayudar en este caso el contrato ágil es a descubrir esta situación lo antes posible e intentar renegociar los términos o incluso cancelar el proyecto.

Y una última pregunta ¿puedo hacer un contrato ágil en un proyecto de coste fijo donde el cliente no ha definido los requisitos? La respuesta es NO, pero es que en estos casos tampoco se debería hacer un contrato tradicional por obra. También hay que aprender a decir que no.

3. Cómo vender TDD

En este punto lo primero que hay que preguntarse es **¿qué es TDD?** o por lo menos ¿qué entiende uno por TDD? Para mi TDD:

- No es una técnica de test.
- No es una técnica de refactorización.
- **Es una técnica de diseño**.

De esta forma TDD es una herramienta o proceso que nos ayuda durante "la fase" de diseño. Por eso los pasos de TDD son: **rojo - verde - refactorizar**. Es decir:

1. Hago el test o ejemplo de código que usa la funcionalidad que necesito desarrollar. Evidentemente en este punto el test falla (rojo) porque no tengo ni siquiera escrito el código.
2. Escribo el códgio mínimo (lucha minimáááá http://www.youtube.com/watch?v=qC72NaYTCzE) para cumplir con el ejemplo del paso 1, vuelvo a ejecutar el test y debería pasar correctamente poniéndose en verde.
3. Si hay código duplicado, clases enormes o algún mal olor, es tiempo de hacer refactoriazaciones (fundamental el libro Refactoing de Martin Fowler).

Es fundamental hacer el test/ejemplo lo primero porque lo que estamos haciendo es un diseño mínimo, es decir vamos a escribir sólo las clases y métodos que necesitamos para cumplir con el ejemplo que hemos escrito. Si invertimos el orden del paso 1 y 2 y hacemos el código antes que el ejemplo ya no será "programación minimáááá", sino que simplemente estaremos haciendo test (esto no es malo, todo lo contrario, es buenisísimo y muy recomendable, pero no es TDD ni nos ayuda en el diseño).

Por todo esto podemos afirmar que Test Driven Development (desarrollo dirigido por pruebas) es realmente un mal nombre para TDD. Sería mucho mejor Example Driven Development (desarrollo dirigido por ejemplo).

Acordaros siempre que una cosa es el diseño y otra cosa son las pruebas. El diseño siempre se hace antes de la implementación, y las pruebas siempre se hacen después de la implementación.

Por último, decir que como **TDD es una herramienta**, la usaremos cuando sea útil, es decir TDD es el medio, no el fin; y nos encontraremos con casos en los que TDD no sea la herramienta adecuada y tengamos que usar otras herramientas, como UML, una servilleta de un bar, hablar con un compañero, pintar en una pizarra, â€¦ (si somos ágiles somos ágiles, y eso significa adaptarse también al uso de herramientas y técnicas).

En cuanto al tema de **cómo vender TDD**, podemos pensar en:

- Vender TDD al jefe.
- Vender TDD al equipo.

En el caso de **vender TDD al jefe** la conclusión podría ser que en vez de preguntar "¿cómo vender TDD al jefe?", se debería preguntar "¿por qué o para qué quiero vender TDD al jefe?". Es decir, realmente me da igual vender o no vender TDD al jefe. Es una herramienta que usará el quipo, que no tiene porque conocer o entender el jefe (¿sabe tu jefe hacer un rebase en git o mercurial? ¿se lo vas a intentar vender?). Evidentemente si lo entiende, será mucho mejor porque facilita la comunicación entre personas.

El caso de **vender TDD al equipo** es el más complicado y el que realmente importa, ya que será precisamente el equipo el que tendrá que usar TDD. En esta situación nos podemos encontrar dos tipos de problemas:

- los miembros que no quieren hacer TDD (casos perdidos, como decía mi Sensei: "no se puede enseñar a quien no quiere aprender"),
- los miembros que si quieren hacer TDD y les encanta, pero que no hacen TDD.

Es este segundo grupo el que realmente me preocupa, ya que es gente convenida e ilusionada con TDD, es gente que te "compra" TDD, pero luego no lo usa. Y ¿por qué? pues la verdad es que no lo se, porque los seres humanos somos así de raros y de complejos; pero hay que reconocer que cuantos de nosotros nos hemos comprado algún "juguetito" con toda la ilusión del mundo, y luego acaba cogiendo polvo en un armario ;)

4. Conclusiones

Esta ha sido mi primera reunión con Madrid Agil y espero que no la última, porque está claro que si quieres avanzar en algo que te gusta no hay nada mejor que juntarte con gente con las mismas inquietudes, y aprender aprender y aprender.

5. Sobre el autor

Alejandro Pérez García, Ingeniero en Informática (especialidad de Ingeniería del Software) y Certified ScrumMaster

Socio fundador de Autentia (Formación, Consultoría, Desarrollo de sistemas transaccionales)

mailto:alejandropg@autentia.com

Autentia Real Business Solutions S.L. - "Soporte a Desarrollo"

http://www.autentia.com

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

» **Registrate** y accede a esta y otras ventajas «

COMENTARIOS

jcarmonaloeches *2010-09-03 - 09:19:06*

Muchas gracias por compartir este conocimiento!