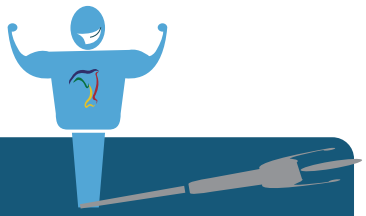
autentia

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...

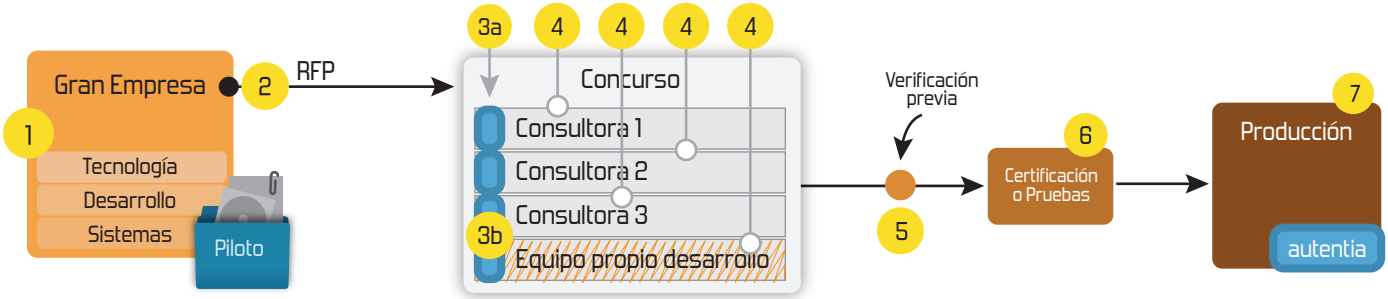
1. Desarrollo de componentes y proyectos a medida



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

Roberto Canales Mora

Creador y propietario de AdictosAlTrabajo.com, Director General de Autentia S.L., **Ingeniero Técnico de Telecomunicaciones y Executive MBA por el Instituto de Empresa 2007.**

Twitter: Seguir a @rcanalesmora 1,414 seguidores

Autor de los Libros: Planifica tu éxito: de aprendiz a empresario y Informática profesional, las reglas no escritas para triunfar en la empresa

Puedes consultar mi CV y alguna de mis primeras aplicaciones (de los 90) aquí

Ver todos los tutoriales del autor

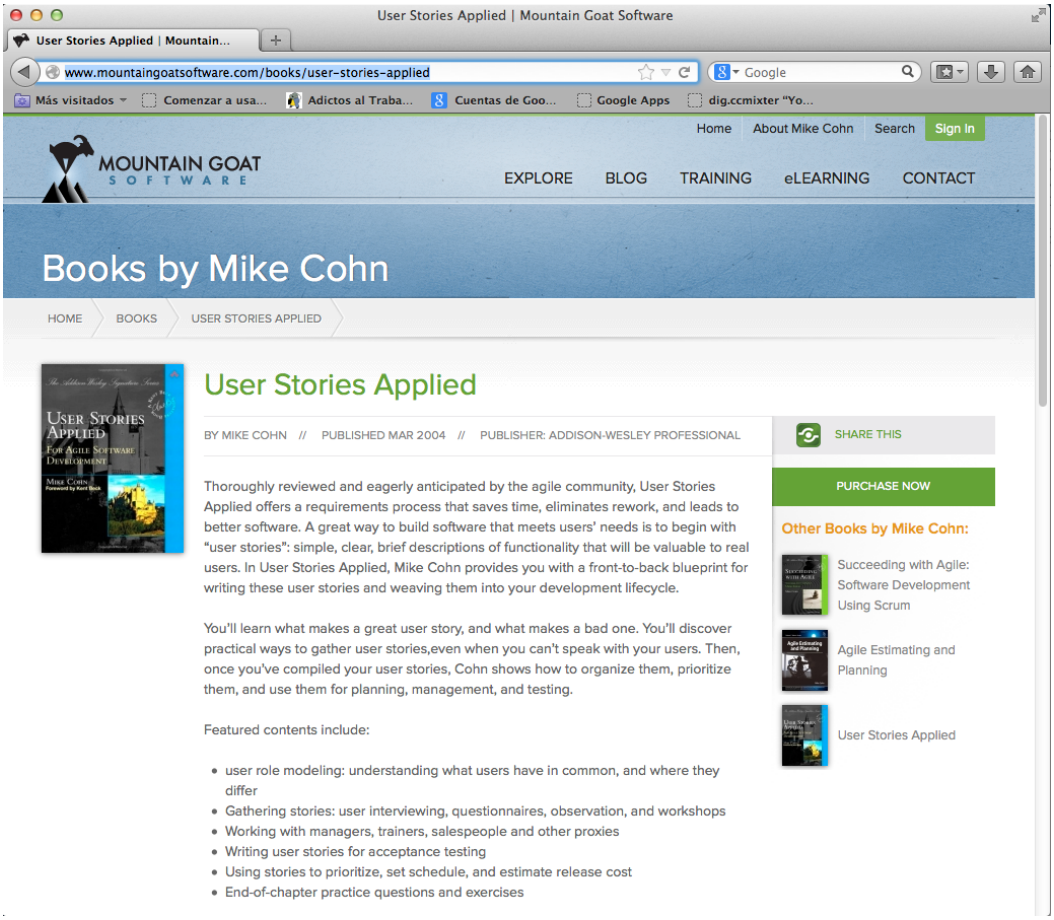
Fecha de publicación del tutorial: 2013-09-20

Tutorial visitado 4 veces Descargar en PDF

Comentando User Stories Applied for Agile Software Development de Mike Cohn

Antes que nada decir que los libros de Mike Cohn son estupendo: sencillos, concretos y bien ordenados. En muchos puntos te quedas con ganas de más.

Podéis encontrar el prólogo, tabla de contenidos y hasta un capítulo fundamental en:
http://www.mountaingoatsoftware.com/books/user-stories-applied



Una de las primeras cosas en las que hay que fijarse es que es una obra con registro de 2004... a esas alturas en España no se si cuanta gente había con este nivel de madurez.

Voy a enumerar las cosas que me han ido gustan (aunque no están en el mismo orden que en el libro). Hasta los resúmenes de los capítulos por el autor son grandiosos por lo que leyéndolos solamente puedes tener una visión de lo vital de la obra (me da casi vergüenza escribir mi propio resumen.. pero ya que estoy)

En el prólogo, Ken Beck cuenta que la construcción de un proyecto plantea problemas porque hay muchos actores que tienen necesidades distintas: Los jefes de proyecto quieren controlar el progreso, los programadores quieren construir, los jefes de producto quieren flexibilidad, los testers quieren métricas, los usuarios usabilidad... un problema tiene muchas perspectivas distintas.

Comenta (haciendo a su vez referencia a Ken Beck) como la realización de unos requisitos por adelantado muy elaborados (a parte de ser imposible) puede matar el proyecto de muchos modos: Cuando son un objetivo en sí mismo, por las imprecisiones del lenguajes, etc. También añadiría que los usuarios no saben muchas veces lo que quieren hasta que lo empiezan a ver. Saturar a un usuario haciendo preguntas sobre cosas derivadas de otras, que todavía no han visto nunca, parece de poco sentido.

Una historia tiene 3 componentes:

- Una descripción escrita usada para planificar y para recordarla.
- Una conversación que sirve para refrescar detalles.
- Unos pruebas que determinan cuando una historias está completa.

Es mejor obtener historias combinando métodos: conversación con usuario, observación, etc. Habla de la importancia de distintos tipos de proxies y de la tentación de los "clientes comerciales" que pueden desviar constantemente el interés del proyecto (y orden de historias) en base a los puntos que creen que pueden conseguirles la próxima venta.

Es preferible tener historias pequeñas (aunque no en exceso) que grandes. Casi cualquier sistema se podría definir en un par de historias (épicas). También es verdad que si el nivel de atomicidad es muy bajo empiezan a aparecen dependencias forzadas entre ellas.

La historias deben ser priorizadas por el usuario de negocio pero debe también considerarse el orden en base a su riesgo o cómo son de complementarias con otras.

Esto va a picar a mucha gente enemiga de medir (o tratar de hacerlo): las historias de usuario no pueden ser priorizadas sin considerar su coste. Lo cuenta de un modo muy graciosos diciendo que su prioridad para las últimas vacaciones eras ir a Tahiti .. hasta que consideró su coste.

Las historias de usuario proporcionan algunas ventajas sobre otras aproximaciones:

- Enfatiza la comunicación verbal.
- Es comprensible tanto para desarrolladores y otros stakeholders.
- Las historias tienen el correcto tamaño para planificar.
- Las historias funcionan con desarrollos interactivos.
- Las historias animan a diferir el detalle hasta tener un mejor conocimiento.

Las historias tienen que ser:

- Independiente: podría poder construirse el cualquier orden y una estimación no afectaría a otra.
- Negociable: Una historia inicialmente muy detallada anulará la conversación y nos separará del cliente.
- Valorable para usuarios y clientes: Aporta valor a negocio, no es un detalle técnico.
- Estimable.
- Pequeñas.
- Testable.

La dificultad de estimar correctamente historias viene por tres razones:

- Lagunas de conocimiento sobre el dominio del problema de los desarrolladores.
- Lagunas de conocimiento sobre la tecnología de los desarrolladores.
- Historias demasiado grandes.

En grandes proyectos, es fácil perdernos. Un modo de empezar a tirar del hilo es que cada usuario defina sus objetivos. Es mejor redactar las historias haciendo referencia a roles específicos. Se puede organizar un juego inicial de roles posibles identificando atributos diferenciadores de cada uno (incluso se puede hacer rol play con características extremas). Los roles posteriormente se pueden consolidar. Si se piensa que puede ayudar, se pude crear algún actor no humano.

Las épicas fallan en dos categorías:

- Historias compuestas.
- Historias demasiado complejas.

Los spikes son buenas soluciones para crear historias de investigación acotando un tiempo para luego poder estimar con certeza la historia principal.

La precisión en la estimación de una historia decrece a medida que incrementa el tamaño de la misma. Del mismo modo, si tenemos muchas historias muy pequeñas, existe la tentación de dar un valor medio a todas ellas prostituyendo las estimaciones. Una buena referencia de tamaño es que se pueda hablar de ella con facilidad en la máquina de café.

Se pueden construir prototipos de baja fidelidad. Sobre el prototipo (prototipar el flujo) se pueden hacer preguntas para identificar historias ausentes:

- ¿Qué es lo siguiente que le gustaría hacer?
- ¿Qué errores se podrían cometer aquí?
- ¿Qué podría confundir al usuario en este punto?
- ¿Qué información adicional el usuario podría necesitar? Ojo con los prototipos porque especificar demasiado precozmente la solución reduce la conversación.

Habla del riesgo de numerar las historias de usuario. Se pierde parte del componente verbal que es lo que les da sentido. Si se tiene la tentación de hacer (lógico para mantener integridad de paneles con herramientas) es conveniente

Últimos Tutoriales del Autor

» Comentando el libro The Leader´s Guide to Radical Management de Stephen Denning

» Comentando Management 3.0, de Jurgen Appelo

» Comentando el libro: Disciplined Agile Delivery (DaD)

» Comentando la AOS 2013

» Comentando el Atlassian Day organizado por Deiser

poner un resumen para no perder su formulación.

Las historias se estimarán preferentemente por puntos. Una vez que se definen las historias, y se estiman por separado, conviene triangular: estimar una respecto a las otras.

A la hora de descomponer una historia en tareas hay criterios útiles:

- Si una tarea es difícil de estimar (incluso si tiene una dependencia a terceros) separar esta tarea del resto de la historia.
- Si una historia tiene partes que pueden hacer desarrolladores en paralelo, partirla.
- Si tenemos ventajas por saber que una parte de una historias esta hecha, descomponerla.

Hay que tener en cuenta el horizonte temporal a la hora de descomponer y detallar las historias. El detalle de las historias debe ser mayor en aquellas que se vayan a afrontar antes. Según escribo esto me entra la risa de los proyectos de toma de requisitos de 20 personas durante 10 meses. Los desarrolladores tienen la tentación de hacer descomposición de historias por descomposición técnica. Esto tiene inconvenientes como que no es comprensible ni útiles desde la perspectiva del usuario. Las historias deben estar lo más cerradas posible. Las historias con palabras genéricas se pueden descomponer.

Si una historia no se puede ejecutar completamente en una iteración: descomponerla en otras historias. Esto ya en más en ejecución del proyecto. Del mismo modo hay veces que las historia son demasiado pequeñas y conviene juntarlas en una.

Los test de aceptación proporcionan unos criterios básicos para saber si una historias está completa, evitando hacer demasiado o demasiado poco esfuerzo. Es importante escribir los test antes que el código. Probar forma parte del proceso. Los test son para encontrar errores. Debemos usar la intuición, conocimiento y experiencias pasada para guiar el esfuerzo de pruebas. El porcentaje de cobertura de código no debe ser una obsesión.

Es necesario esperar varias iteraciones antes de poder estimar la velocidad del equipo. La experiencia en el refinamiento de las historias hará que sean más homogéneas. No se deben contabilizar resultados parciales sino solamente aquellas que cumplan todos los criterios de aceptación.

La principal diferencia entre casos de uso e historias (si bien las dos están pensados para entregar valor a cliente) es el tamaño. Una historias está acotada a lo que uno o varios desarrolladores pueden hacer en unos días. Se parecerían más a un escenario o colaboración de un caso de uso aunque estos últimos también suelen ser de mayor ámbito.

Hay varios malos olores que nos pueden ayudar a identificar problemas en la definición de historias:

- Muy pequeñas.
- Interdependientes.
- Incorporadas por desarrolladores sin petición de cimiente.
- Demasiado detalle.
- Incluyen el interfaz de usuario demasiado pronto.
- Definidas en detalle cuando no se van a implementar todavía.
- Dividir demasiado las tareas.
- El cliente tiene problemas a la hora de priorizar.
- El cliente no escribe/prioriza las historias.

El libro tienes muchas más cosas interesantes como explicación de las historias de usuario con Scrum, concepto de XP, ejemplos, etc. Desde luego me parece una obra indispensable en la bibliografía de un agilista. Si ya complementas esta lectura con otras obras tendrás una visión más completa y actualizada: os enseño algunas con las que he estado recientemente:

- The Leaders Guide to Radical Management de Stephen Denning
- Agile Management de Angel Medinilla
- Disciplined Agile Delivery (DaD)
- Management 3.0, de Jurgen Appelo

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

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