15 de febrero de 2017
Bogotá D.C.

William Sierra william.sierra@exsis.com.co
Carlos Cardona carlos.cardona@exsis.com.co
Ervid Molina ervid.molina@exsis.com.co

The Clean Coder Cap. 7 Pruebas de aceptación.

El papel del desarrollador profesional involucra un rol de comunicación, así como un rol de desarrollo. Además un desarrollador profesional es cuidadoso, al cerciorarse de que su comunicación con otros miembros del equipo y el área comercial, sea exacta y saludable.

REQUERIMIENTOS DE COMUNICACIÓN

Uno de los problemas de comunicación más comunes entre los desarrolladores y el área comercial son los requerimientos. Los analistas de negocio dicen lo que creen que necesitan, y luego los desarrolladores construyen lo que creen que describe el negocio. Al menos eso es lo que se supone que funciona. En realidad, la comunicación de requerimientos es extremadamente difícil, y el proceso está lleno de errores.

Pero se ha encontrado y se ha aprendido una poderosa lección sobre cómo los clientes realmente descubren lo que necesitan. La perspectiva que se tiene inicialmente sobre las características del negocio, no suelen sobrevivir porque los clientes están reconsiderando constantemente sus necesidades a medida que van teniendo contacto con el software.

PRECISIÓN PREMATURA

Tanto el área comercial como los desarrolladores están tentados a caer en la trampa de una precisión prematura. El área comercial quiere saber exactamente lo que va a conseguir antes de autorizar un proyecto. Los desarrolladores quieren saber exactamente qué se supone que se debe entregar antes de estimar el proyecto. Ambas posiciones quieren una precisión que simplemente no se puede lograr, y están a menudo dispuestos a desperdiciar una fortuna tratando de alcanzarla.

EL PRINCIPIO DE INCERTIDUMBRE

El problema es que los requerimientos son unos en el papel y son otros en el sistema cuando éste entra en ejecución. Cuando el área comercial observan que es lo que especificaron en el sistema, se dan cuenta que eso no era para nada lo que querían. Una vez que analizan si los requerimientos se satisfacen en producción, tienen una

mejor idea de como realmente quieren el sistema, y usualmente esa no es la manera como lo están viendo.
Por ello es necesario brindar nueva información al área comercial, con el fin de contemplar al sistema en su totalidad. Al final, cuanto más precisos sean sus requerimientos, menos relevantes se vuelven a medida que se implementa el sistema.

ANSIEDAD EN LA ESTIMACIÓN

Todo desarrollador sabe que debe estimar el tiempo para crear un sistema y que existen herramientas que requieren precisión; si no considera esto será consumido por la ansiedad. Las buenas estimaciones contemplan altas márgenes de variación. Además de esto el principio de incertidumbre hace que se discuta exhaustivamente la precisión a corto plazo. Esto generará cambios en los requerimientos.
Los desarrolladores profesionales entienden que las estimaciones pueden y deben hacerse basándose en requerimientos de baja precisión y las reconocen como verdaderas estimaciones. En complemento, los desarrolladores profesionales incluyen barras de error con sus estimaciones para que el área comercial entienda la incertidumbre.

AMBIGÜEDAD TARDÍA

La solución a la precisión prematura es aplazar la precisión tanto como sea posible. Los desarrolladores profesionales no fabrican un requerimiento hasta que están seguros de que necesitan desarrollarlo. Sin embargo, eso puede conducir a otro problema: ambigüedad tardía. "Una ambigüedad en un documento de requerimientos representa un argumento entre las partes interesadas".
Por supuesto, no se necesita un argumento o un desacuerdo para crear ambigüedad. A veces los stakeholders (partes interesadas) simplemente asumen que sus lectores interpretan los requerimientos de la misma manera. Puede ser perfectamente claro para ellos en su contexto, pero significan algo completamente diferente para los desarrolladores que lo leen.
Es responsabilidad de los desarrolladores profesionales y de los stakeholders asegurarse de que toda ambigüedad se elimine de los requerimientos.

PRUEBA DE ACEPTACIÓN

Vamos a definir las pruebas de aceptación como pruebas escritas producto del trabajo conjunto entre los stakeholders y los desarrolladores con el fin de definir qué requerimientos están listos para probar.

- **LA DEFINICIÓN DE "TERMINADO"**

Una de las ambigüedades más comunes que enfrentamos como profesionales que trabajan en software es la ambigüedad de "terminado".
Los desarrolladores profesionales tienen una sola interpretación para algo “terminado”: significa que está terminado. Es decir, que el código ha sido escrito, que todas las pruebas aplicadas fueron aprobadas , que cumple los estándares de calidad y que los stakeholders lo han aceptado.
Los desarrolladores profesionales dirigen la definición de sus requerimientos hasta llegar a pruebas de aceptación automatizadas. Es necesario colaborar con los stakeholders y con QA (control de calidad) para asegurar que estas pruebas automatizadas son una especificación terminada del requerimiento.

- **COMUNICACIÓN**

El propósito de las pruebas de aceptación es comunicar con claridad y precisión. Para llegar allí, todas las partes involucradas deben entender el comportamiento del sistema. Los desarrolladores profesionales tienen la responsabilidad de trabajar con los stakeholders y los testers para asegurarse de que todos conozcan lo que está a punto de construirse.

- **AUTOMATIZACIÓN**

Las pruebas de aceptación siempre deben ser automatizadas. Hay un lugar para las pruebas manuales en otras partes del ciclo de vida del software, pero este tipo de pruebas nunca deben ser manuales. La razón es simple: costo.

Los desarrolladores profesionales asumen la responsabilidad de su parte en asegurar que las pruebas de aceptación sean automatizadas. Existen muchas herramientas de códigos abiertos y comerciales que facilitan la automatización de las pruebas de aceptación. FitNesse, Pepino, cuke4duke, estructura de robot y Selenio, por mencionar sólo algunas.

- **TRABAJO EXTRA**

Algunos desarrolladores creen que las pruebas de aceptación son sinónimo de trabajo extra, pero no lo son realmente. Estas pruebas representan la especificación del sistema en sus componentes más pequeños. Especificar en este nivel de detalle es la única manera en que los desarrolladores pueden saber lo que "terminado" significa. Así que no considere estas pruebas como un trabajo extra. Visualícelos como ahorradores masivos de tiempo y dinero. Estas pruebas le evitarán la implementación incorrecta del sistema y le permitirá saber cuándo haya terminado.

- **¿QUIÉN ESCRIBE PRUEBAS DE ACEPTACIÓN Y CUÁNDO?**

En un mundo ideal, los stakeholders junto con el control de calidad colaborarian para escribir estas pruebas, y los desarrolladores revisarian su consistencia. En la práctica, los stakeholders raramente tienen el tiempo o la disposición para sumergirse a un nivel de detalle requerido. Así que a menudo delega la responsabilidad en los analistas de negocio, control de calidad, o incluso los desarrolladores. Si resulta que los desarrolladores deben escribir estas pruebas, entonces tenga en cuenta de que el desarrollador que escribe la prueba no es el mismo que implementa la característica probada.

Además, la perspectiva para crear las pruebas varía mucho entre analistas y miembros de calidad. Si tenemos en cuenta el principio de precisión tardía, tales pruebas deberían tomar un tiempo en ser creadas y ejecutadas.

- **EL PAPEL DEL DESARROLLADOR**

Se enfoca en conectar las pruebas de aceptación al sistema, y luego hacer funcionar esas pruebas. No hay que preocuparse demasiado por escenarios o accesorios, ya que las fallas del sistema serán recogidas por las pruebas, para mejorar su funcionamiento.

- **NEGOCIACIÓN DE LA PRUEBA Y ACTITUD PASIVO-AGRESIVA**

Como desarrollador profesional, es su deber negociar con el autor de la prueba para que construya una mejor prueba. Lo que nunca debe hacer es tomar la opción pasivo-agresiva y considerar: "Bueno, eso es lo que dice la prueba, así que eso es lo que

voy a hacer". Recuerde, que como profesional es su trabajo ayudar a su equipo a crear el mejor software posible. Esto significa que todo el equipo involucrado debe estar atento a los errores y trabajar juntos para corregirlos.

- **PRUEBAS DE ACEPTACIÓN Y PRUEBAS UNITARIAS**

Las pruebas de aceptación no son pruebas unitarias. Las pruebas unitarias son escritas por desarrolladores para desarrolladores. Son documentos de diseño formal que describen la estructura de nivel más bajo del sistema y el comportamiento de su código. No están dirigidas a otro actor diferente al desarrollador.

Las pruebas de aceptación están escritas por el área comercial para ellos mismos (aunque el desarrollador termina escribiendolas). Son documentos de requisitos formales que especifican cómo debe comportarse el sistema desde el punto de vista del negocio. Su objetivo son los desarrolladores y analistas de negocio.

Estas pruebas no son redundantes, aunque en ocasiones prueben las mismas cosas. Tienen diferentes mecanismos y objetivos. Juntas tienen como propósito documentar el diseño, la estructura y comportamiento del sistema.

- **GUIS Y OTRAS COMPLICACIONES**

GUIs (Graphic user interface, Interfaz gráfica de usuario)

Es difícil especificar GUIs de antemano. Se puede hacer, pero rara vez se hace bien. La razón es que la estética es subjetiva y por lo tanto volátil. Esto hace que sea un reto escribir pruebas de aceptación para GUIs. El truco es diseñar el sistema para que pueda tratar la GUI como si fuera una API en lugar de un conjunto de botones, deslizadores, cuadrículas y menús.

Se debería desacoplar el GUI y las reglas de negocio, para reemplazarlos con pequeños módulos que los pongan a prueba.

Existe un principio de diseño llamado Principio de Responsabilidad Única (SRP). Este principio establece que debe separar las cosas que cambian por diferentes razones, y agrupar las cosas que cambian por las mismas razones.

- **PRUEBAS A TRAVÉS DE LA INTERFAZ CORRECTA**

Mejor aún es escribir pruebas que invocan las características del sistema subyacente a través de una API real en lugar de utilizar un GUI. Esta API debe ser la misma API utilizada por la GUI. Así que escriba sus pruebas de reglas de negocio para pasar por una API justo debajo de la GUI. Mantenga las pruebas GUI a un mínimo. Es frágil, porque la GUI es volátil. Cuantas más pruebas GUI tenga, menos posibilidades tendrá de mantenerlas.

- **INTEGRACIÓN CONTINUA**

Asegúrese de que todas sus pruebas de unidad y aceptación se ejecutan varias veces al día en un sistema de integración continua. Este sistema debe ser activado por un sistema de control de código fuente.

Contener la presa

Es muy importante mantener las pruebas de CI funcionando en todo momento. Nunca deben fallar. Si fracasan, entonces todo el equipo debe detenerse, y centrarse en reparar las pruebas rotas para pasar de nuevo. .

CONCLUSIÓN

La comunicación sobre los detalles es difícil. Esto es especialmente cierto para desarrolladores y stakeholdres que se comunican sobre los detalles de una aplicación. Es muy fácil que cada parte agite sus manos y asuma que la otra parte entiende. Con demasiada frecuencia ambas partes están de acuerdo en que entienden y se van con ideas completamente diferentes.

La única manera que consideramos de eliminar eficazmente los errores de comunicación entre los desarrolladores y los stakeholders es escribir pruebas de aceptación automatizadas. Estas pruebas son formales al ejecutarse. Ellas son completamente inequívocas, y no pueden salir de la sincronización con la aplicación. Representan el mejor documento de requerimientos posible.