Tests

Tests - Conceptos

Difference between Error, Fault, Defect and Failure

Publicado el 6 de marzo de 2015

Ivan Luizio Magalhães | Seguir

Increasing Profit Through Process and Customer Experience I...

19 0 0

"A clever person solves a problem. A wise person avoids it."

Albert Einstein

https://www.linkedin.com/pulse/difference-between-error-fault-defect-failure-ivan-luizio-magalh%C3%A3es

Tests - Propósito

- **Verification** - The verification process confirms that the software meets its technical specifications. A specification is a description of a function in terms of a measurable output value given a specific input value under specific preconditions. A simple specification may be along the line of a SQL query retrieving data for a single account against the multi-month account-summary table must return these eight fields <list> ordered by month within 3 seconds of submission.

- **Validation** - The validation process confirms that the software meets the business requirements. A simple example of a business requirement is after choosing a branch office name, information about the branch's customer account managers will appear in a new window. The window will present manager identification and summary information about each manager's customer base: <list of data elements>. Other requirements provide details on how the data will be summarized, formatted and displayed.

- **Defect Finding** - A defect is a variance between the expected and actual result. The defect's ultimate source may be traced
to a fault introduced in the specification, design, or development (coding) phases.

Tests - Propósito

- **Error:** A human action that produces an incorrect result. The mistakes made by programmer is knowns as an ‘Error’. This could happen because of the following reasons: some confusion in understanding the requirement of the software; some miscalculation of the values; or/and misinterpretation of any value, etc.

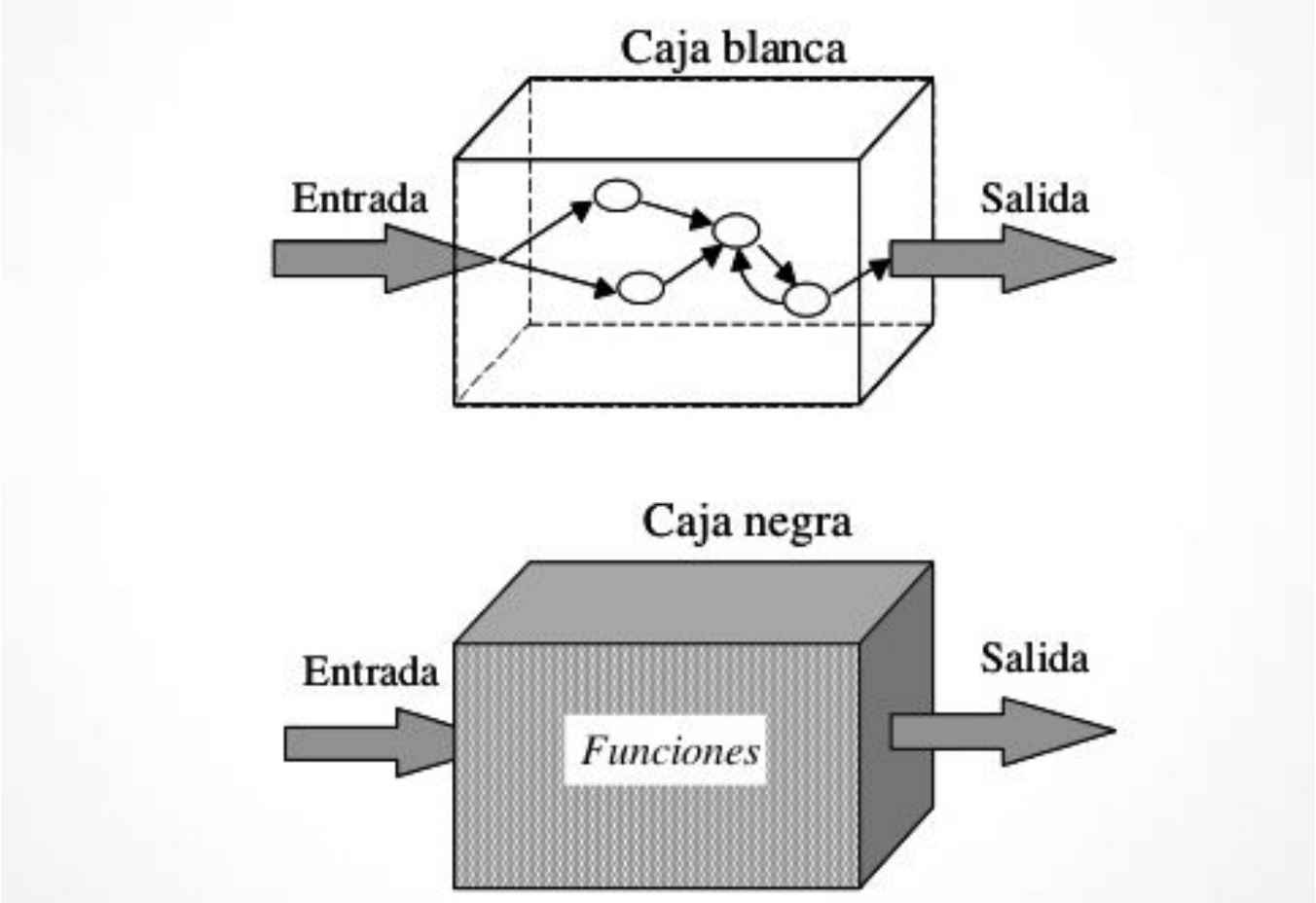
- **Fault:** A manifestation of na 'Error' in software. Faults are also known colloquially as defaults or bugs.

- **Defect:** The departure of a quality characteristic from its specified value that results in a product not satisfying its normal usage requirements. The 'Error' introduced by programmer inside the code are known as a 'Defect'. This can happen because of some programatical 'Error'.

- **Failure:** Failure is a deviation of the software from its intended purpose. If under certain circumstances these 'Defects' get executed by the tester during the testing then it results into the 'Failure' which is known as software 'Failure'.

Tests - Clasificaciones

http://es.slideshare.net/jabenitez88/8realizacion-de-pruebas-14981770



Tests - Clasificaciones

http://www.javiergarzas.com/2014/07/tipos-de-pruebas-10-min.html

Pruebas unitarias

- Con ellas probamos las unidades del software, normalmente métodos.
- Por ejemplo, escribimos estas pruebas para comprobar si un método de una clase funciona correctamente de forma aislada.
- Las pruebas unitarias corresponden a la visión de los desarrolladores
- Las dependencias complejas o interacciones con el exterior se gestionan realizando mocks (objetos ya programados con los datos que se espera recibir).
    - Aunque estés probando un método que realice una serie de cosas y al final mande un correo electrónico a través de un servidor de correo, en una prueba unitaria no tienes que probar que el correo se ha mandado correctamente.
    - Buenas pruebas unitarias no irían contra una base de datos, por ejemplo, sino que simularían la conexión.
- Si una prueba unitaria falla, sabes que es por un problema en el código.
- Para automatizar y realizar este tipo de pruebas se utilizan framework de tests, por ejemplo unittest

Tests - Clasificaciones

http://www.javiergarzas.com/2014/07/tipos-de-pruebas-10-min.html

Pruebas integración

- En este caso probamos cómo es la interacción entre dos o mas unidades del software.
- Este tipo de pruebas verifican que los componentes de la aplicación funcionan correctamente actuando en conjunto.
- Siguiendo con el caso anterior, las pruebas de integración son las que comprobarían que se ha mandado un email, la conexión real con la base de datos etc.
- Este tipo de pruebas son dependientes del entorno en el que se ejecutan. Si fallan, puede ser porque el código esté bien, pero haya un cambio en el entorno.
- Por ejemplo, también podríamos usar unittest para realizar pruebas de integración

Tests - Clasificaciones

http://www.javiergarzas.com/2014/07/tipos-de-pruebas-10-min.html

Pruebas funcionales

- El objetivo de las pruebas funcionales es comprobar que el software que se ha creado cumple con la función para la que se había pensado.
- En este tipo de pruebas lo que miramos, lo que nos importan, son las entradas y salidas al software. Es decir, si ante una serie de entradas el software devuelve los resultados que nosotros esperábamos.
- Aquí solo observamos que se cumpla la funcionalidad, no comprobamos que el software esté bien hecho, no miramos el diseño del software. Estudiamos el software desde la perspectiva del cliente, no del desarrollador.
- Por eso este tipo de pruebas entran dentro de lo que se llaman **pruebas de caja negra**.

Tests - Clasificaciones

http://www.javiergarzas.com/2014/07/tipos-de-pruebas-10-min.html

Pruebas de Carga

- Las pruebas de carga son un tipo de prueba de rendimiento del sistema.
- Ver cómo se comporta el sistema ante X usuarios que entran concurrentemente a la aplicación y realizan ciertas transacciones.

Pruebas de Estrés

- Este es otro tipo de prueba de rendimiento del sistema.
- El objetivo de estas pruebas es someter al software a situaciones extremas, intentar que el sistema se caiga, para ver cómo se comporta, si es capaz de recuperarse o tratar correctamente un error grave.

Pruebas de Aceptación

Por último, las pruebas de aceptación se realizan para comprobar si el software cumple con las expectativas del cliente, con lo que el cliente realmente pidió.

Tests - Clasificaciones

- **HAY MUCHAS MÁS CLASIFICACIONES**
- Hay mucho humo en esta disciplina.
- Tomar recomendaciones como de quien viene.
- No subestimar los tests.
- El TDD es probablemente de lo mejor que nos dio la ing de software.

Cómo informar de fallos de forma efectiva

http://www.chiark.greenend.org.uk/~sgtatham/bugs-es.html

Introducción

- Existe una razón por la cual el soporte técnico se ve como un empleo horrible: los malos informes de fallo.
- El objetivo de un informe de fallo es permitir al programador ver el programa fallando ante sus ojos
- Hay que dejar muy claro qué son **hechos reales** ("Estaba con la computadora y ocurrió esto") y qué son **especulaciones** ("*Creo* que esto puede ser el problema").
- Se puede excluir las especulaciones pero nunca los hechos.
- Cuando informas de un fallo, lo haces porque quieres que el fallo se arregle, el fallo se arreglará antes si les ayudas proporcionándoles toda la información que necesitan.

Cómo informar de fallos de forma efectiva

http://www.chiark.greenend.org.uk/~sgtatham/bugs-es.html

No funciona.

- Dale al programador algo de crédito en cuanto a inteligencia básica: si el programa no funcionase en absoluto, se hubieran dado cuenta.
- Por tanto, o bien estás haciendo algo diferente de lo que ellos hacen, o tu entorno es diferente del suyo. Necesitan información; proporcionar esa información es el objetivo de un informe de fallo. Más información casi siempre es mejor que menos.
- Muchos programas, particularmente programas gratuitos, publican una lista de fallos conocidos.
- Si tienes un fallo conocido, probablemente no merezca la pena informar de él nuevamente.
- Pero si piensas que tienes más información que la que aparece en el informe de fallo, puede que sean capaces de arreglar el fallo más fácilmente si les das información que todavía no tenían.
- Si el programa viene con su propio conjunto de consejos para informar de fallos, léelos.
- Si no estás informando de un fallo sino simplemente pidiendo ayuda para utilizar el programa, deberías indicar en qué otros sitios has buscado ya la respuesta a tu pregunta. Esto permitirá que el programador sepa dónde espera la gente encontrar la respuesta, de forma que pueden hacer que la documentación sea más fácil de usar.

Cómo informar de fallos de forma efectiva

http://www.chiark.greenend.org.uk/~sgtatham/bugs-es.html

Muestrame.

- Deja que miren cómo arrancas la máquina, que miren cómo ejecutas el software, que miren cómo interactúas con el software y que miren lo que el software realiza como respuesta a tus entradas.
- Los programadores saben en qué partes confiar, y saben qué partes pueden tener más fallos. Saben intuitivamente lo que tienen que buscar.
- Quizás esto no sea suficiente. Quizás decidan que necesitan más información, y te pidan que les muestres lo mismo una vez más.
- Una vez que puedan ver el problema, normalmente podrán seguir solos a partir de ese punto para intentar arreglarlo.

Cómo informar de fallos de forma efectiva

http://www.chiark.greenend.org.uk/~sgtatham/bugs-es.html

Enséñame a mostrar por mí mismo.

- El objetivo del ejercicio es hacer que sean capaces de reproducir el problema.
- Así que diles exactamente lo que hiciste.
- Si es un programa gráfico, diles qué botones pulsaste y en qué orden los pulsaste. Si es un programa que se ejecuta escribiendo una orden, múestrales de forma precisa la orden que usaste.
- Cuando sea posible, debes proporcionar una transcripción de la sesión,
- Dale al programador toda la información que se te ocurra.
- Si el programa lee de un fichero, probablemente necesites enviarles una copia del fichero.
- Si el programa habla con otro computador a través de una red, probablemente no puedas enviarles una copia de ese computador, pero puedes al menos decir qué clase de computador es y, si es posible, qué software ejecuta.

Cómo informar de fallos de forma efectiva

http://www.chiark.greenend.org.uk/~sgtatham/bugs-es.html

A mí me funciona. ¿Qué está mal?

- Si le das a un programador una larga lista de entradas y acciones y ellos lanzan su propia copia del programa y no ocurre nada malo, entonces es que no les has dado información suficiente.
- Posiblemente el fallo no ocurre en todos los computadores; tu sistema y su sistema puede diferir de algún modo.
- Quizás hayas malentendido lo que el programa supuestamente hace, y ambos estáis mirando exactamente la misma pantalla y tú piensas que está mal y ellos saben que está bien.
- Así que describe también lo que ocurrió. Diles exactamente lo que viste. Diles por qué piensas que lo que viste está mal; aún mejor, diles exactamente lo que tú esperabas ver.
- Si viste mensajes de error entonces dile al programador, de forma clara y precisa, cuáles eran. ¡*Son* importantes!
- En este momento, el programador no está intentando arreglar el problema: sólo están intentando encontrarlo.

Cómo informar de fallos de forma efectiva

http://www.chiark.greenend.org.uk/~sgtatham/bugs-es.html

Y entonces intenté…

- Hay un montón de cosas que puedes hacer cuando aparece un fallo o un error. Muchas de esas cosas empeoran el problema.
- Cuando algo va mal, inmediatamente deja de hacer nada. No toques ningún botón en absoluto. Mira la pantalla e intenta encontrar cosas fuera de lo normal, y recuérdalo o anótalo.
- Si logras salir del problema, bien cerrando el programa afectado o bien reiniciando la computadora, sería bueno intentar hacer que el problema ocurra de nuevo.
- A los programadores les gustan los problemas que pueden reproducir más de una vez.

Cómo informar de fallos de forma efectiva

http://www.chiark.greenend.org.uk/~sgtatham/bugs-es.html

Creo que el modulador de taquiones tiene la polarización incorrecta.

- Nadie dice: "Doctor, necesito que me prescriba hidroyoyodina." La gente sabe que no debe hacer eso con un doctor: le describes los síntomas, las molestias y achaques y dolores y erupciones y fiebres, y dejas que el doctor haga el diagnóstico.
- Siempre establecer los síntomas es mejor que proporcionar tu propio diagnóstico
- Igualmente, enviar modificaciones del código para arreglar el problema es un suplemento útil en un informe de fallo, pero no un sustituto adecuado para el mismo.

Cómo informar de fallos de forma efectiva

http://www.chiark.greenend.org.uk/~sgtatham/bugs-es.html

Vaya, lo hacía hace un momento.

- Los problemas fáciles son aquellos en los que la realización de una secuencia simple de acciones hacen que surja el problema. Demasiados problemas no funcionan así.
- La mayoría de los fallos intermitentes no son verdaderamente intermitentes. La mayoría de ellos tienen alguna lógica en alguna parte.
- El programador querrá saber todo lo que puedas contarle del problema. Pruébalo en otra máquina, quizás. Pruébalo dos o tres veces y mira la frecuencia con la que falla.
- Si falla cuando estás trabajando en algo serio pero no cuando estás intentando demostrarlo, puede que sean los tiempos grandes de ejecución o ficheros grandes lo que lo hagan fallar. Intenta recordar todos los detalles que puedas acerca de lo que estabas haciendo cuando el programa falló, y
- Si encontraste algún tipo de patrón, menciónelo.
- Incluso si es información probabilística (como "falla con más frecuencia si estoy ejecutando el Emacs")
- Más importante, el programador querrá estar seguro de si es un auténtico fallo intermitente o un fallo específico de esa máquina. Querrán conocer un montón de detalles acerca de tu computador.

Cómo informar de fallos de forma efectiva

http://www.chiark.greenend.org.uk/~sgtatham/bugs-es.html

Entonces cargué el disco en Windows...

- Escribir de forma clara es esencial en un informe de fallo. Si el programador no puede entenderte, quizás sea mejor no decir nada.
- La mayoría de informes confusos provienen de hablantes nativos de inglés que suponen que les entenderé aunque no hagan ningún esfuerzo por ser claros o precisos.
- **Sé específico.** Si puedes hacer lo mismo de dos maneras, di cuál usaste. "Seleccioné Cargar" puede significar "Pulsé sobre Cargar" o "Pulsé Alt+L". Di cuál usaste. A veces importa.
- **Se prolijo.**
- **Ten cuidado con los pronombres y los verbos sin sujeto.** No uses referencias como "la ventana"
- **Lee lo que has escrito.**

Cómo informar de fallos de forma efectiva

http://www.chiark.greenend.org.uk/~sgtatham/bugs-es.html

Resumen.

- El primer objetivo de un informe de fallo es permitir que el programador vea el fallo por sí mismo.
- Si el primer objetivo falla El segundo objetivo del informe de fallo es describir lo que falló.
- Cuando el computador haga algo inesperado, detente
- Por todos los medios, intenta diagnosticar el problema tú mismo si crees que puedes, pero si lo haces, informa también de los síntomas igualmente.
- Prepárate para proporcionar información extra si el programador la necesita.
- Escribe de manera clara. Di lo que quieres decir, y asegúrate de que no se puede malinterpretar.
- Por encima de todo, sé preciso.