02/03/2016

Jose Antonio Ruiz Millán

Ejercicios FIS

Ingeniería Informática UGR

Relación 1.1:

Ejercicio 3:

Ejercicio 4:

Mito 1: Mito del cliente.

Mito → Para comenzar a escribir programas, es suficiente el enunciado general de los objetivos, podremos entrar en detalles más adelante.

Realidad → Aunque no siempre es posible tener el enunciado exhaustivo y estable de los requerimientos, un "planteamiento de objetivos" ambiguo es una receta para el desastre. Los requerimientos que no son ambiguos se desarrollan sólo por medio de una comunicación eficaz y continua entre el cliente y el desarrollador.

Mito 2: Mito del profesional.

Mito → Una vez que escribimos el programa y hacemos que funcione, nuestro trabajo ha terminado.

Realidad → Alguien dijo alguna vez que "entre más pronto se comience a 'escribir el código', más tiempo tomará hacer que funcione". Los datos de la industria indican que entre 60 y 80% de todo el esfuerzo dedicado al software ocurrirá después de entregarlo al cliente por primera vez.

Ejercicio 7:

Ejercicio 9:

- Nuestra mayor prioridad es **satisfacer al cliente** mediante la **entrega temprana y continua** de software con valor.
    - Intenta solucionar el contacto con el cliente, ya que es de vital importancia tener al cliente informado de la evolución del software ya que él nos puede verificar si va todo correcto o en alguna parte ha habido algún mal entendimiento y corregir el error.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la **conversación cara a cara.**
    - Todos sabemos que hablando se solucionan las cosas, debemos intentar hablar los problemas de desarrollo directamente con las personas responsables del tema ya que será más rápido y fácil solucionarlos que mediante correo u otra forma de contacto.
- Los **responsables de negocio y los desarrolladores** trabajamos **juntos** de forma cotidiana durante todo el proyecto.
    - Si todos los integrantes en el proyecto trabajan juntos, agiliza el trabajo y se trabaja de forma más ordenada y clara.

- La atención continua a **la excelencia técnica y al buen diseño** mejora la Agilidad.
    - Claramente si cada paso que damos lo damos lo más perfecto posible evitamos tener que corregir errores en una etapa más tardía y tener que replantear lo que ya teníamos acabado.
- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de **tiempo más corto posible.**
    - Claramente debemos satisfacer al cliente lo máximo posible, y una de las tareas a cumplir es la rapidez de la creación del software final. Con esto se pretende no alargar la espera del cliente hasta incluso pudiendo llegar a no realizar la entrega por pasar demasiado tiempo que el cliente no le interesa esperar.

Relación 1.2:

Ejercicio 1:

1.- Error en la programación o prueba del sistema software:

El 4 de junio de 1996 la Agencia Espacial Europea lanzo el cohete Ariane 5. Un error de programación en el módulo de gestión provoco la autodestrucción del chete 37 segundos después del despliegue.

2.- El diseño del software o del hardware:

La sonda de la NASA Mars Climate Orbiter, que fue lanzada a Marte el 11 de diciembre de 1998 y 286 días después sobrevoló el planeta rojo a 57 kilómetros de su superficie en vez de los 150 previstos, lo que provocó que se destruyera en atmósfera marciana. El culpable del error en la trayectoria de la sonda fue el 'software' informático basado en la Tierra. El fallo destruyó un proyecto de 327 millones de dólares.

3.- El estudio de problemas:

Con la popularización de Internet, surgieron los primeros fallos importantes relativos a la manera que tenían de interactuar los sistemas operativos existentes con la Red. El más grave **fue el "ping of death"**, que duró de 1995 a 1996. Un "ping" es una señal que puede lanzarse un ordenador a otro para comprobar que ésta "rebota" y vuelve, comprobando en primer lugar que la dirección destino existe y está operativa, y en segundo el tiempo que tarda en realizar el trayecto. Sin embargo, si se modificaba el código de este paquete de información deliberadamente, se podía hacer que el ordenador destino se colgase sin remisión. Aunque los principales afectados fueron computadoras con sistemas operativos de Microsoft, este tipo de ataque también afectó a equipos Macintosh y a los basados en Unix.

4.- Un mal uso:

Definitivamente, los peores errores son los que causan víctimas, y éste causó **ocho víctimas mortales y más de veinte personas quedaron seriamente afectadas**. Un sistema de radioterapia es controlado por un programa diseñado por la compañía estadounidense Multidata Systems International, que calcula la dosis de radiación adecuada en cada caso. Para proteger el tejido sano, se pueden dibujar en una pantalla hasta 4 bloques virtuales para indicar en qué zona se tiene que irradiar. Los doctores panameños quieren utilizar cinco bloques, pero el programa no

se lo permite, así que descubren que pueden simular la configuración de cinco bloques dibujando uno solo con un agujero en el medio. El problema es que el programa, dependiendo de cómo se dibuje el agujero (en sentido de las agujas del reloj o al contrario), dobla la dosis de radiación recomendada, causando la muerte o graves daños al paciente. En el caso judicial contra el centro, los doctores son acusados de negligencia, ya que debían comprobar manualmente que las dosis recomendadas por el programa eran las correctas, pero queda claro que el software no debería haber dejado tanta libertad al usuario para equivocarse con maquinaria potencialmente letal.

5.- La seguridad:

Un fallo de seguridad en el sistema de screeners de Hollywood expone decenas de películas inéditas. Mas información.

6.- Un deficiente análisis del riesgo:

Apagón del 2003 en EEUU

Un fallo en el nuevo software de control, que, si se colgaba, enviaba una señal al resto de centrales para que pudieran reaccionar. El problema era que esta señal hacía que las máquinas que la recibían se colgaran y reiniciaran, enviando a su vez el fatídico mensaje a cada vez más centrales de distribución. El resultado fue que se bloquearon más de 100 plantas eléctricas y más de 50 millones de hogares estuvieron sin electricidad hasta que se detectó el error.

Ejercicio 2:

Claramente el peor de todos es el que tiene que calcular la dosis de radiación adecuada para cada persona, ya que se produjeron víctimas mortales.

En mi caso ninguno tiene que ver con la industria armamentista.

Con la industria aeroespacial están relacionados tanto el caso 1 como el caso 2.

Con la industria sanitaria está relacionada el caso 4 que causo víctimas mortales.

Con las finanzas podrían estar relacionados tanto el de la fuga de las películas de Hollywood como el apagón de EEUU ya que supondría un coste adicional.

Ejercicio 4:

Debemos tener muy claro lo que vamos a realizar y sobre todo para quien va dedicado el software ya que no es lo mismo crear un software para la gestión de una biblioteca a crear un software para el ejército o para un hospital donde puede haber vidas en juego. Por lo tanto dependiendo de la importancia de nuestro software tenemos que hacer miles y miles de pruebas antes de entregarlo al usuario final sobre todo si como he dicho anteriormente este software puede conllevar una perdida humana o económica.