Jose Miguel Hernández García
2º B

Ejercicio 4 Relación 1.1

Mitos del gestor

¿Por qué debemos cambiar nuestra forma de desarrollar software? Si estamos haciendo el mismo tipo de producto ahora que hace diez años…
La realidad de este mito es que aunque el dominio de la aplicación pueda ser igual siempre, la demanda de mayor productividad y calidad, y el papel crítico del software en objetivos comerciales han aumentado bastante. Por lo que es necesario adaptarse a los nuevos tiempos.

Si fallamos en la planificación, podemos añadir más programadores y recuperar el tiempo perdido.
Sin embargo, la realidad es que durante el proceso de software, añadir más personal puede retrasar más el proyecto, los desarrolladores han de añadirse de forma ordenada y planificada. Además si sacamos a gente de otros proyectos, provocará que estos otros se retrasen. No por aumentar el tamaño del equipo de desarrollo se van a obtener mejores resultados (Concepto de Horda mongoliana).

Mitos de los clientes

Una declaración general de los objetivos es suficiente para comenzar a escribir los programas, y podemos dar los detalles más adelante.
Una mala definición inicial es la primera causa por la que se produce un trabajo inútil, por lo que todo debe quedar claro antes de comenzar a desarrollar software puesto que sino se podría provocar una gran pérdida de tiempo y recursos.

Los requisitos del proyecto cambian continuamente, pero los cambios pueden acomodarse fácilmente porque el software es flexible.
Es cierto que los requisitos cambian y esto es algo que en cierto modo es inevitable, pero el impacto de este cambio puede variar según el momento en que estos cambios se produzcan.

Ejercicio 7 Relación 1.1

Gestión de una biblioteca
La gestión de una biblioteca podría realizarse mediante una base de datos donde se almacenen los ejemplares de libros que la biblioteca tenga y los socios pertenecientes a la biblioteca, por lo que el esfuerzo no sería mucho dado que no es algo de excesiva dificultad. Por otro lado si que es cierto que habría que mantener las bases de datos en constante actualización y mantenimiento, dado que siempre aumenta el número de libros y de socios.
El coste de la creación del software tampoco sería muy elevado por lo comentado anteriormente. También es cierto que en el caso de que una persona quiera retirar un libro de la biblioteca y el software no esté listo aún, es posible apuntar en un papel, como antaño, el número de socio y el libro en cuestión, por lo que no es necesario apresurarse mucho en la entrega del software, eso si, tampoco hay que excederse del plazo de entrega.
El modelo que más puede adecuarse al proyecto en cuestión es el modelo incremental donde una vez entregada la primera versión del software, ya se esté planeando una segunda versión que intente mejorar las carencias y defectos de la primera, y así hasta obtener una versión lo más óptima posible.

Ejercicio 9 Relación 1.1

Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
Este principio nos dice que la “Alianza Ágil” trata de evitar los retrasos a la hora de entregar el software, manteniendo satisfechos a los clientes además, no solo entregando una primera versión de este software, sino también realizando un adecuado mantenimiento de este y actualizándolo de forma continua.

La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

Con este principio, la "Alianza Ágil" nos dice que tratan de fomentar la simplicidad para así evitar problemas como la repetición de código o el trabajo ineficiente, a favor de realizar el mínimo trabajo posible.

Ejercicio 1 Relación 1.2

1. **Marine Bugs Out ->** Fallo en la programación del software
2. **Hartford Coliseum Collapse ->** Deficiente análisis del riesgo
3. **CIA Gives the Soviets Gas ->** Deficiente análisis del riesgo
4. **World War III … Almost ->** Fallo en el diseño del software
5. **Medical Machine Kills ->** Fallo en el diseño del software
6. **Wall Street Crash ->** Mala documentación o uso
7. **AT&T Lines Go Dead ->** Fallo en el diseño del software
8. **Patriot Fails Soldiers ->** Fallo en el diseño del software
9. **Pentium Fails Long Division ->** Incorrecto estudio de problemas
10. **Ariane Rocket Goes Boom ->** Fallo en la programación del software
11. **Skynet Brings Judgement Day ->** Deficiente análisis del riesgo
12. **Mars Climate Crasher ->** Fallo en la programación del software
13. **Disastrous Study ->** Fallo en la programación del software
14. **British Passport to Nowhere ->** Deficiente análisis del riesgo
15. **Y2K ->** Fallo en la programación del software
16. **Dot-Bomb Collapse ->** Deficiente análisis del riesgo
17. **Love Virus ->** Fallo de seguridad
18. **Cancer Treatment to Die For ->** Fallo en la programación del software
19. **EDS Drops Child Support ->** Fallo en el diseño del software
20. **FBI's Trilogy Terminated ->** Incorrecto estudio de problemas

Ejercicio 2 Relación 1.2

1. **¿Cuál es el intruso?**
   El intruso es Love Virus

2. **¿Cuál consideras que es el peor de todos? ¿por qué?**
   El peor de todos podría ser la tercera guerra mundial dado que de haber sucedido tal hecho podría haber afectado a toda la humanidad, pudiendo causar daños gravísimos e incluso provocando que estos daños arrasen con todo el planeta.

3. **¿Cuáles están relacionados con la industria armamentística?**
   World War III… Almost
   Patriot Fails Soldiers
   Skynet Brings Judgement Day

4. **¿Cuáles están relacionados con la industria aeroespacial?**
   Marine Bugs Out
   Ariane Rocket Goes Boom
   Mars Climate Crasher

5. **¿Cuáles están relacionados con la industria sanitaria?**
   Medical Machine Kills
   Disastrous Study
   Cancer Treatment to Die For

6. **¿Cuáles están relacionados con las finanzas?**
   Wall Street Crash
   Dot-Bomb Collapse
   EDS Drops Child Support

Ejercicio 4 Relación 1.2

Pienso que es necesario tanto conocer como informarse acerca de los problemas que puede provocar un fallo de seguridad, o un fallo en el diseño, en la programación… dado que estos problemas pueden desencadenar problemas que afecten a una gran cantidad de personas, o provocar desastres como la caída de Wall Street o la casi 3ª Guerra Mundial, por lo que siempre hay que tratar de realizar un trabajo lo más eficiente posible, comprobando las veces que sea necesario que todo está en orden y está hecho de forma correcta antes de que cualquier problema pueda salir a la luz y causar algún desastre. Es por ello que una de las soluciones podría ser encargarle a un equipo específico que busque los posibles fallos del sistema, los solucione e intenten que no vuelvan a ocurrir antes de que el sistema software salga a la venta o se exponga al público, tratando de presentar al público una versión lo más óptima posible del software en cuestión donde se eviten estos y otros posibles fallos.