Calidad del software

Adrián Riesco

Universidad Complutense de Madrid, Madrid, Spain

Auditoría, Calidad y Fiabilidad Informáticas 2022/23

Calidad del software

Contenidos

- En este tema veremos qué es la calidad, cómo medirla y cómo garantizarla.
- Para ello, veremos en primer lugar una breve introducción con las definiciones de los conceptos más importantes que se verán en el resto del tema.
- Después de ello veremos qué debemos tener en cuenta a la hora de preparar un proyecto de software.
- Una vez empezado el proyecto, veremos cómo integrar el control de la calidad con el desarrollo del software.
- También veremos cómo controlar la calidad de la documentación y del mantenimiento del software.

Contenidos

- Además veremos métodos para controlar que el proyecto se ajusta a los plazos y el presupuesto.
- Por último, hablaremos sobre algunas métricas y estándares importantes en el control de calidad.
- Sin embargo, la calidad del software es un tema muy complejo y muchas cuestiones deben dejarse fuera.
- Otros temas interesantes que podrían ser tratados en un curso más exhaustivo son:
  - Garantía de calidad cuando personal externo participa en el desarrollo.
  - Formación y certificación del personal.
  - Aplicación de acciones preventivas y correctivas.
  - El coste de garantizar la calidad.

Contenidos

¿Qué es el software?

- Primero debemos plantearnos qué es el software.
- Según IEEE,

Definición

Software son los programas, procedimientos y posiblemente la documentación asociada y los datos relacionados al funcionamiento de un sistema informático.

- El resto de organizaciones oficiales dan una definición similar.

¿Qué es el software?

- Los programas, es decir, el código, son obviamente necesarios, dado que es lo que se va a ejecutar.
- Por *procedimientos* no entendemos las "funciones" implementadas en los programas, sino a la forma en el que se debe ejecutar el código.
- Es decir, los procedimientos son las instrucciones sobre cómo instalar y configurar.

¿Qué es el software?

- La documentación es necesaria para:
  - Continuar con el desarrollo.
  - Depurar el código existente.
  - Ayudar al usuario.
- La documentación del desarrollo permite una cooperación eficiente dentro del equipo de desarrollo y la inspección de los resultados.
- La documentación para el mantenimiento describe las tareas de cada módulo, de tal manera que se pueda depurar.
- La documentación para el usuario (como el "manual del usuario") describe las opciones y cómo usarlas.
- Los dos primeros tipos de documentación están relacionados.

¿Qué es el software?

- Los datos no se refieren solo a los ficheros de configuración.
- Es también necesario proporcionar información sobre las pruebas (tests) a los que se ha sometido el software.
- Esto sirve para comprobar el funcionamiento de nuevas versiones y para informar de posibles errores.

Fallos, defectos y errores software

- Entenderemos como fallos software (software failures) los comportamientos erróneos que se producen al ejecutar un programa.
- El origen de estos fallos son los errores software (software errors) del programador.
- Un error puede ser *gramatical*, en el código, o un error *lógico* en la interpretación de uno de los requisitos del cliente.
- Estos errores dan lugar a defectos software (software faults), comportamientos erróneos del sistema.
- Es posible que un defecto no produzca un fallo, porque los argumentos que lo revelan no son utilizados nunca o porque se "neutraliza" con otros defectos.
- Podemos entender los fallos software como los defectos software que son detectados.

Fallos, defectos y errores software

Esta imagen ilustra la relación entre errores, defectos y fallos:[1]

[1] Gracias a Luis Enrique Ortiz Orué Flores por proporcionar la imagen.

Fallos, defectos y errores software

Ejemplo: Hotel

Es posible que, al desarrollar el software para un hotel, nuestro cliente nos pida que no se permita hacer una reserva a un cliente que haya dejado a deber más de 50 euros entre todas sus estancias anteriores.
Sin embargo, el programador encargado de la aplicación escribió un 0 de más, por lo que ese error genera un defecto.
En cualquier caso, como al hacer la reserva se pide al cliente una tarjeta de crédito y se comprueba antes de hacer la reserva final, es imposible que algún cliente deje dinero a deber, y por tanto este defecto nunca llega a detectarse como un error.

Discusión

Discusión

Un programador en nuestra empresa insiste en decir que, dado solo una pequeña parte de los errores software acaban siendo errores software no es necesario dedicar demasiados recursos a prevenir y eliminar dichos errores.

- *¿Estáis de acuerdo con esta opinión?*
- *¿Qué ocurre cuando se siguen estas ideas (algo que ocurre en algunos lugares, aunque quizás por distintas razones)?*

Clasificación de errores software según su causa

- Podemos clasificar los errores software en función de su causa en:
  - Errores de código.
  - Errores de procedimiento.
  - Errores de documentación.
  - Errores de datos.
- Todos estos errores son errores humanos.

Clasificación de errores software según la etapa del proceso de desarrollo

1. Errores en la definición de requisitos.
   - Estos errores se deben en general a deficiencias en la definición de requisitos dada por el cliente.
   - Puede ser una definición errónea o incompleta de un requisito.
   - La ausencia de un requisito.
   - La existencia de un requisito innecesario.

Clasificación de errores software según la etapa del proceso de desarrollo

2. Errores de comunicación entre cliente y desarrollador.
   - Malentendidos en los requisitos iniciales.
   - Malentendidos en los cambios de los requisitos.
   - Malentendidos en las respuestas del cliente a los problemas de diseño encontrados durante el desarrollo.

Clasificación de errores software según la etapa del proceso de desarrollo

3. Cambios intencionados respecto a los requisitos software.
   - Reutilización de código sin comprobar adecuadamente que cumple los requisitos.
   - Omisión de parte de la funcionalidad por problemas de tiempo o presupuesto.
   - Optimizaciones en el software que no han sido requeridas ni aprobadas por el cliente. Estas optimizaciones pueden parecer irrelevantes al programador pero generar errores más tarde.

Clasificación de errores software según la etapa del proceso de desarrollo

4. Errores en el diseño lógico.
   - La traducción de los requisitos dada por el cliente normalmente está especificada en lenguaje natural.
   - Cuando los ingenieros los traducen a un lenguaje más formal se pueden producir errores.
   - Asignación equívoca de algoritmos.
   - Errores de secuencia.

Ejemplo: Cobrador de deudas

Nos han solicitado un sistema en el que, cuando una persona tiene deudas, es notificado por correo certificado. Una vez se han enviado 3 notificaciones y no se ha recibido respuesta se notifica al director del departamento financiero, que decidirá si se lleva a juicio.
Sin embargo, el diseñador puede introducir un error si rompe la secuencia e informa que hay que llevar a juicio al deudor sin consultar con el departamento financiero.

Clasificación de errores software según la etapa del proceso de desarrollo

4. Errores en el diseño lógico (cont.).
   - Errores en la definición de los límites.

Ejemplo: Descuentos

El cliente puede pedir que un cliente tenga descuento si alguien compra más de 3 veces al mes, pero el diseñador indica que se hace descuento para $c \geq 3$, con c el número de veces que ha comprado este mes.

   - Omisión de estados.

Ejemplo: Software para climas extremos

El cliente pide diseñar un sistema de tiempo real que depende de la combinación entre temperatura y presión. Al diseñar la aplicación puede faltar un estado que el usuario ha definido de manera laxa, como por ejemplo qué ocurre a $120^{\circ}C$ y entre 6 y 8 atmósferas de presión.

Clasificación de errores software según la etapa del proceso de desarrollo

4. Errores en el diseño lógico (cont.).
   - Omisión de las definiciones de los estados erróneos.

Ejemplo: Venta de entradas

El cliente solicita que no se pueden vender más de 10 entradas, y en caso contrario mostrar un mensaje de error. Si el diseñador simplemente indica que se venden como máximo 10 entradas, el programa fallará en el resto de ocasiones.

   - Aunque estos casos pueden parecen inverosímiles, hay que tener en cuenta que la formalización de requisitos complejos a partir de lenguaje natural es una tarea complicada.

Clasificación de errores software según la etapa del proceso de desarrollo

5. Errores en el código.
6. Errores de coherencia entre la implementación y la documentación.
   - Si no se documentan adecuadamente las interfaces, los equipos a cargo de usar dicha interfaz la utilizarán erróneamente.
   - Si el programador de las funciones mal documentadas deja la empresa, su sustituto tendrá problemas para entender el código.
   - El equipo a cargo de comprobar que la implementación se ajusta a los requisitos tendrá problemas.
   - El equipo de pruebas (testing) puede obtener resultados erróneos para funciones correctas.
   - El equipo de depuración puede introducir errores, en lugar de solucionarlos.

Clasificación de errores software según la etapa del proceso de desarrollo

7. Defectos en el proceso de prueba.
   - Cuando el testing es incompleto se dan por correctas partes erróneas de código.
   - Esto puede llevar a entorpecer el proceso de depuración, dado que se presta menos atención a las funciones que se consideran correctas.
   - Estos errores se pueden dar tanto por negligencia como por problemas de tiempo, que solo permiten comprobar la funcionalidad mínima solicitada por el usuario y no los potenciales problemas.

Clasificación de errores software según la etapa del proceso de desarrollo

8. Errores de procedimiento.
   - Los procedimientos guían al usuario en cada paso del proceso.
   - Son especialmente importantes cuando la aplicación genera varios resultados que son posteriormente utilizados, y que tienen sentido por sí mismos.

Ejemplo: Venta de materiales

Una empresa de venta de materiales ofrece un descuento del 5 % a sus clientes, a los que se cobra una vez al mes. En concreto, el descuento se ofrece a aquellos clientes que han comprado productos por más de un millon de euros en los últimos 12 meses. Sin embargo, los directivos han decidido no aplicar el descuento a los clientes que hayan devuelto productos por valor de al menos el 10 % de sus compras en los últimos 3 meses. El sistema de cobro no está centralizado (una de las principales razones para precisar los procedimientos), por lo que las facturas de cada tienda se procesan de manera independiente.

Clasificación de errores software según la etapa del proceso de desarrollo

Ejemplo: Venta de materiales (cont.)

El procedimiento correcto consiste, al principio de cada mes, en:

- Para la oficina central, (i) recopilar las ventas hechas en cada tienda; (ii) calcular las ventas acumuladas para cada cliente; (iii) calcular el porcentaje de productos devueltos en los últimos 3 meses; (iv) calcular la lista de clientes que cumplen los requisitos para el descuento y distribuirla a las tiendas.
- Para las tiendas, (i) calcular las ventas hechas en el último mes y (ii) usar la lista recibida para aplicar los descuentos en el mes entrante.

En general cualquier otro procedimiento, como trabajar con años naturales o trimestres, llevará a resultados erróneos.

Clasificación de errores software según la etapa del proceso de desarrollo

9. Errores de documentación.
   - Ya hemos comentado que hay errores de documentación que dificultan el desarrollo de productos.
   - Los errores de documentación también pueden afectar al cliente.
     - Cuando se olvida documentar alguna función.
     - Cuando se documentan aplicaciones que finalmente no se implementaron, o que cambiaron a lo largo del desarrollo de la aplicación.
     - Cuando las instrucciones son incompletas.

Calidad del software: definición

- En primer lugar, tenemos la definición dada por el IEEE:

Definición (Calidad del software – IEEE)

La calidad del software es:

1. *El grado en el que un sistema, componente o proceso cumple con los requisitos.*
2. *El grado en el que un sistema, componente o proceso cumple con las expectativas o necesidades del cliente.*

- Para que ambos puntos se cumplan es necesario que el cliente sea capaz de transmitir de manera efectiva el sistema que desea, lo que no es cierto en la práctica.

Calidad del software: definición

- Otras definiciones siguen la misma línea:

Definición (Calidad del software – Crosby, 1979)

Calidad significa adecuarse a los requisitos.

Definición (Calidad del software – Juran, 1988)

1. *Consideramos como calidad aquellas características de los productos que cumplen las necesidades del cliente y por tanto le producen satisfacción.*
2. *La calidad es la ausencia de deficiencias.*

- En esta última definición, ¿qué es una deficiencia?

Calidad del software: definición

Definición (Calidad del software – Pressman, 2000)

La calidad del software se define como la adecuación a los requisitos de funcionalidad y rendimiento explícitamente expresados, a los estándares de desarrollo explícitamente documentados y a las características implícitas que se esperan del software profesional.

- Esta definición tiene 3 puntos importantes:
  - La funcionalidad de la herramienta, que se define esencialmente en función de las salidas.
  - Los estándares de calidad que se indiquen en el contrato.
  - Las Buenas Prácticas en Ingeniería del Software (Good Software Engineering Practices – GSEP).
- Esta definición será útil para evaluar la calidad del software.

Discusión

Discusión

La defincción de calidad de Pressman es claramente beneficiosa para el programador, porque cualquier cosa que no esté correctamente especificada y acabe resultando en un error es culpa del cliente.

- *¿Hasta qué punto puede el cliente especificar su sistema? (es decir, ¿por qué debería tener conocimientos de informática cualquier persona que necesite una aplicación informática?).*
- *¿Cómo puede el programador ayudar en esta tarea?*

Garantía de calidad: definición

- De nuevo, comenzamos con la definición del IEEE sobre la garantía (assurance) de calidad del software:

Definición (Garantía de calidad – IEEE)

La garantía de calidad del software es:

1. *Un patrón sistemático de todas las acciones necesarias para proporcionar la confianza necesaria para asegurar que un producto cumple los requisitos técnicos necesarios.*
2. *Un conjunto de actividades diseñado para evaluar el proceso mediante el cual los productos se desarrollan o manufacturan. Se complementa con el control de calidad.*

Garantía de calidad: definición

- La definición del IEEE hace énfasis en:
  - Cumplir con los requisitos.
  - El proceso (técnico) de desarrollo.
- Sin embargo, no tiene en cuenta otros aspectos:
  - Las necesidades de mantenimiento del software.
  - Relación con los plazos y los presupuestos (¿es igual de bueno el producto si ha costado la cantidad esperada que si ha costado el triple?).

Garantía de calidad: definición

- Teniendo en cuenta estas ideas, podemos dar una definición más completa:

Definición (Garantía de calidad – definición extendida)

La garantía de calidad del software es: Un conjunto de acciones sistemático necesario para proporcionar la confianza adecuada para asegurar que el proceso de desarrollo del software o el proceso de mantenimiento de un producto software se adhiere a los requisitos técnicos y funcionales establecidos, así como a los requisitos administrativos de cumplir los plazos y los presupuestos.

- Esta definición se ajusta a las ideas presentadas en el estándar ISO 9000.

Garantía de calidad vs. Control de calidad

- Al hablar de calidad del software, las palabras recurrentes son "garantía" y "control".
- El control de la calidad es el conjunto de técnicas para evaluar la calidad de un cierto producto.
- Dado que evalúan un producto, este tiene que estar acabado.
- La garantía de calidad trata de minimizar el coste de asegurar la calidad durante el proceso de desarrollo.
- Para ello busca y corrige errores en las primeras etapas del desarrollo.
- Además, para garantizar la calidad, se debe estudiar el producto una vez terminado.
- Por tanto, podemos considerar el control de calidad como la última fase del proceso de garantía de calidad.

Garantía de calidad: Objetivos

- Podemos distinguir dos categorías de objetivos: durante el desarrollo y durante la vida útil del producto (mantenimiento).

1. Desarrollo de software (orientado al proceso):
   - Asegurar con un nivel aceptable de confianza que el software satisfará los requisitos técnicos funcionales.
   - Asegurar con un nivel aceptable de confianza que el software se adecuará a los plazos y el presupuesto.
   - Iniciar y controlar las acciones para la mejora del desarrollo de software y de las actividades de garantía de calidad.
   - Este último punto indica que se debe mejorar el proceso de desarrollo sin disparar el coste de los procesos de garantía de calidad, que haría su aplicación inútil.

Garantía de calidad: Objetivos

2. Mantenimiento del software (orientado al producto)
   - Asegurar con un nivel aceptable de confianza que el mantenimiento del software satisfará los requisitos técnicos funcionales.
   - Asegurar con un nivel aceptable de confianza que el mantenimiento del software se adecuará a los plazos y el presupuesto.
   - Iniciar y controlar las acciones para la mejora del mantenimiento del software y de las actividades de garantía de calidad.
   - De nuevo, no se debe reducir los costes de mantenimiento a base de disparar el coste en el proceso de garantía de calidad.

Garantía de calidad e Ingeniería del software

- Como se puede deducir de todo lo que hemos visto hasta el momento, la calidad del software está íntimamente relacionada con la Ingeniería del software.
- Básicamente, la Ingeniería del software es el proceso que debemos aplicar para asegurarnos de desarrollar y mantener software de manera sistemática.
- Por tanto, la Ingeniería del software es la base de la que parte la calidad del software, su control y su garantía.
- A partir de ella, añadiremos mecanismos para mejorar y abaratar el software.
- Es por tanto imprescindible la colaboración entre los equipos de Garantía de calidad y de Ingeniería del software.

Contenidos

Factores de calidad del software: Introducción

- Hemos visto hasta ahora que la especificación de los requisitos por parte del usuario es esencial para desarrollar correctamente el software.
- En esta parte del tema justificaremos la necesidad de documentos exhaustivos que permitan al programador desarrollar la aplicación correctamente y satisfaciendo al usuario.
- Para ello, veremos cómo podemos asegurar que dichos requisitos son lo más correctos posibles.
- La idea básica es identificar los factores (aspectos o áreas) que se deben concretar.

La necesidad de documentos exhaustivos

- Es habitual que un programa funcione bien en la mayoría de (o en todas) las ocasiones para los que fue pensado.
- Sin embargo, ligeras modificaciones o un uso imprevisto hace que fallen.
- Esto se debe en general a que el usuario definió qué quería que hiciese el software, pero no definió su uso en el resto de casos...
- ...y el programador tampoco le preguntó.
- Por ello, es necesario que los requisitos sean algo más que las ideas del cliente. Es necesario que estén en un documento con una estructura y unos contenidos claros.

Clasificación de los requisitos dentro de los factores de calidad

- El modelo clásico de los factores de calidad del software lo propuso McCall en 1977.
- Consta de 11 factores.
- Hay 3 categorías en las que se clasifican estos factores:
  - Factores de operación del producto: Corrección, Confiabilidad, Eficiencia, Integridad, Usabilidad.
  - Factores de revisión del producto: Mantenibilidad, Flexibilidad, *Testabilidad*.
  - Factores de transición del producto: Portabilidad, Reusabilidad, Interoperabilidad.
- Veremos cada uno de ellos por separado.

Corrección

- Los requisitos de corrección se definen en una lista compuesta por las salidas que el usuario espera del sistema.
- La idea de salida incluye distintos aspectos:
  - La idea/concepto de qué debe ser la salida (e.g. la temperatura, los beneficios, el saldo de la cuenta, . . .).
  - La completitud de la respuesta (e.g. ¿el saldo de la cuenta incluye los planes de pensiones o las hipotecas?)
  - La exactitud necesaria (e.g. no es lo mismo medir la temperatura en una oficina que en la sala de conservación de un museo).
  - El tiempo de respuesta.
  - La actualidad de la respuesta (e.g. en caso de producirse un terremoto, ¿cuándo se debe medir la potencia?).

Corrección

Ejemplo (Cajeros automáticos)

¿Qué factores de corrección le pedimos al software de un cajero?

- *La salida es el saldo actual.*
- *Debe incluir el dinero en la cuenta corriente y los depósitos, y nada más.*
- *La respuesta tiene 2 decimales.*
- *Se presenta con la última hora de acceso, por si otra persona autorizada está actualizando la cuenta.*
- *No debe tardar más de 2 segundos en mostrar la información.*

Confiabilidad

- Indica la tasa máxima de error que se le permite al sistema.
- Puede ser diferente según las funciones.

Ejemplo (Hospital)

Una unidad de monitorización del corazón debe fallar menos de una vez cada 20 *años.*

Ejemplo (Test de embarazo)

Un test de embarazo desechable puede fallar 1 *de cada* 20 *veces.*
Un test de embarazo en un hospital puede fallar 1 *de cada* 1000 *veces.*

Eficiencia

- Los requisitos de eficiencia indican los recursos necesarios para ejecutar todos los requisitos software.
- Los aspectos más importantes son el procesador, la memoria y la velocidad de los buses/red.
- Puede incluir información sobre los máximos requeridos al hardware (e.g. se necesitará que puede funcionar de manera ininterrumpida por un máximo de 2 semanas, con pausas de 1 día).
- También se deben indicar los requisitos de batería, si el equipo no es fijo.

Eficiencia

Ejemplo (Videojuego)

¿Qué requisitos de eficiencia suelen tener los videojuegos?

- *En general, suelen tener unos mínimos de procesador y disco duro.*
- *Si quieres mejorar los gráficos, necesitas mejorarlos a los requisitos recomendados.*
- *Además puede ser necesaria una tarjeta gráfica.*
- *Si quieres jugar en red, necesitas una conexión de una cierta velocidad para poder hacerlo (que además se puede comprobar antes de conectarte).*

Integridad

- Los requisitos de integridad se refieren a seguridad.
- Es decir, evitar que personas no autorizadas lo usen, o para distinguir diferentes niveles de usuarios.
- Es posible que, en algunos casos, parte de la información se comparta y solo varíen las funciones.

Ejemplo (Sistemas de Información Geográfica)

Los sistemas GIS permiten ver y añadir cierto tipo de información (como comentarios en Google Maps), pero no permiten modificar los mapas más que a los administradores.

Usabilidad

- La usabilidad se refiere a los requisitos para formar empleados capaces de usar el software.

Ejemplo

Será mucho más caro formar a alguien para que opere el software de una central nuclear que para usar una caja registradora.

Mantenibilidad

- Los requisitos de mantenibilidad determinan el esfuerzo necesario, por parte de usuarios y personal de mantenimiento, para identificar los fallos software, corregirlos, y comprobar el éxito de las correcciones.
- Los requisitos de este factor se refieren a la modularidad del sistema, la documentación interna y al manual del programador, entre otras cosas.

Ejemplo

Requisitos de mantenibilidad típicos son:

1. *El tamaño de una función no debe ser mayor de* 30 *instrucciones.*
2. *El programa se debe ajustar a los estándares y directivas de la compañía.*

Flexibilidad

- Los requisitos de flexibilidad se encargan de los recursos necesarios para la adaptación del software.
- Incluyen los requisitos de personal necesarios para personalizar el software para usuarios con distintos perfiles en un área común.
- También las mejoras periódicas del software (las típicas actualizaciones de los sistemas operativos o de cualquier programa).

Flexibilidad

Ejemplo (Club deportivo)

Un club deportivo nos ha pedido que desarrollemos un software para ellos. Es necesario que guarde los datos de los deportistas, que imprima los calendarios, las listas de convocados y las estadísticas de cada jugador.

- *Tiene que ser válido para cualquier deporte. Si el club decide participar en un nuevo deporte o abandonar uno en activo, tiene que ser posible modificar fácilmente los parámetros.*
- *Hay que tener también en cuenta que las categorías por edades por cada deporte pueden variar, así que el usuario debe ser capaz de modificarlo con poco esfuerzo.*

Testabilidad

- Por un lado, los requisitos de testabilidad tienen que proporcionar elementos para ayudar en las pruebas del sistema (por ejemplo, proporcionar resultados intermedios que no son requeridos por el cliente).
- También incluyen comprobaciones sobre el sistema (por ejemplo, se comprueba que los ficheros de configuración existen, que hay suficiente memoria y que tiene un tarjeta gráfica adecuada).

Portabilidad

- Los requisitos de portabilidad indican los recursos necesarios para adaptar el software a otros entornos, como puede ser otro tipo de hardware, de sistema operativo, etc.
- Estos requisitos permiten usar el mismo software básico en distintas situaciones o usarlo simultáneamente en distintas plataformas.

Ejemplo (Videojuegos)

La mayoría de los juegos que no están programados por las propias compañías encargadas del hardware (e.g. Nintendo, Sony y Microsoft) hacen sus juegos multiplataforma.
Sin embargo, en general no es posible que usuarios de un mismo juego en diferentes plataformas puedan jugar juntos online.

Reusabilidad

- La reusabilidad se refiere al uso de módulos desarrollados en un proyecto anterior en el proyecto actual.
- También se pueden referir a preparar un módulo del proyecto actual para su uso en el futuro.
- Reutilizando software se espera ahorrar recursos, acortar los plazos y aumentar la calidad.
- La calidad se espera porque suponemos que los módulos se comprobaron y depuraron en el pasado, por lo que no deben fallar si se usan adecuadamente.
- La reutilización del software se ha descrito ya en algunos estándares.

Reusabilidad

Ejemplo (Hotel)

Nos han encargado hacer el software para el spa de un hotel. En dicho spa pueden entrar tanto los clientes del hotel como los abonados al spa. Además, resulta que hace unos meses este mismo hotel nos encargó un sistema similar para acceder a la piscina, por lo que podemos (y deberíamos) reutilizar parte del código que desarrollamos en ese momento. En concreto, podríamos reutilizar:

- *El lector de tarjetas.*
- *La gestión de renovaciones de abonos.*
- *¿Alguno más?*

Interoperabilidad

- La interoperabilidad se refiere al desarrollo de interfaces para trabajar con otros módulos o con otros programas.
- Por tanto, estos requisitos pueden simplemente indicarse con una lista de programas/módulos con los que queremos que nuestro programa interactúe.

Ejemplo (Hospital)

Podemos esperar que el monitor de actividad de un hospital se conecte con los dispositivos para comprobar el pulso o el ritmo de la respiración.

Actualización de los factores

- Estos factores se actualizaron años después de la propuesta de McCall en 1977.
- Las dos actualizaciones que vamos a comentar conservan todos los factores iniciales excepto el de testabilidad.
- Entre las dos variantes se añadieron 5 nuevos factores:
  - Verificabilidad.
  - Expansibilidad.
  - Seguridad.
  - Manejabilidad.
  - Supervivencia.

Verificabilidad y expansibilidad

- Los requisitos de verificabilidad definen las características de diseño y programación que permiten verificar el diseño y la programación.
- En general estos requisitos se refieren a la modularidad, simplicidad y seguimiento de las directrices de documentación y programación.
- Los requisitos de expansibilidad se refieren a los esfuerzos futuros para mejorar el servicio o la usabilidad.
- Estos requisitos no se diferencian mucho del de flexibilidad.

Seguridad

- Los requisitos de seguridad buscan eliminar las condiciones peligrosas que resultan de un error software.
- Estos errores pueden resultar en reacciones inapropiadas a situaciones comprometidas o a evitar las señales de alarma cuando el software detecta las situaciones peligrosas.

Ejemplo (Planta química)

En una planta química un programa controla el ácido en función de los cambios de presión y temperatura.
Los requisitos de seguridad nos indican que se deben definir las posibles situaciones peligrosas, especificando qué tipo de alarmas deben activarse.

Manejabilidad

- Los requisitos de manejabilidad se refieren a las herramientas de administración que se encargan de las modificaciones de software durante el desarrollo y el mantenimiento del producto.
- Estos requisitos incluyen configuraciones, cambios de procedimientos, etc.

Ejemplo

Software de control de versiones.

Supervivencia

- La supervivencia se refiere a la continuidad del servicio.
- Indican el tiempo mínimo permitido entre errores y el tiempo máximo permitido para que el sistema se recupere.
- Aunque se puede referir a cualquier función, se suele aplicar solo a funciones críticas.
- Parte de estos requisitos ya están recogidos en el factor de confiabilidad.

Contenidos

El contrato

- Obviamente, debemos evitar firmar "malos contratos".
- En nuestro contexto, un "mal contrato" no es uno en el que el cliente "paga poco".
- Un mal contrato es aquel que tiene una definición de requisitos laxa y plazos y presupuestos poco realistas.
- Por lo tanto, uno de los primeros objetivos de la garantía de calidad es conseguir un buen contrato.

El contrato

- El proceso de revisión del contrato tiene dos fases (aunque evidentemente también dependerá de las directivas de cada empresa concreta):
  - Revisión del borrador de la propuesta antes del envío al cliente.
  - Revisión del contrato antes de firmarlo.
- En la primera fase se revisan las bases de la propuesta: los requisitos (con todos los detalles adicionales que se necesiten), estimaciones de costes y recursos, y la existencia de contratos con otras empresas o subcontratas.
- En la segunda fase se revisan todos los cambios alcanzados durante las negociaciones previas.

Revisando la propuesta: objetivos

La revisión de la propuesta tiene los siguientes objetivos:

1. Los requisitos del clientes han sido descritos y documentados.
   - Aunque existen los llamados documentos RFP (Request for Proposal, solicitud de propuesta), en general son demasiado imprecisos para el desarrollo.
   - Por ello, se debe pedir detalles adicionales al cliente. Cada aclaración debe registrarse en un documento por separado que debe ser aprobado tanto por el cliente como por la empresa.

Revisando la propuesta: objetivos

2. Se han estudiado diversas alternativas para desarrollar el proyecto.
   - A menudo, el equipo de desarrollo propone una manera de resolver el problema propuesto por el cliente y no se estudian otras alternativas que podrían resultar más sencillas (y por tanto más rápidas y baratas) e igual de efectivas.
   - Este estudio se refiere, sobre todo, a investigar si es posible reutilizar módulos o subcontratar a un programador con conocimiento específico dentro del área en la que nos encontramos.

Revisando la propuesta: objetivos

3. Se han especificado los aspectos formales entre el cliente y la empresa.
   En este caso se deben definir ciertas formalidades, que incluyen:
   - La manera de comunicarse con el cliente.
   - Qué productos deben entregarse.
   - El proceso de aprobación del producto final.
   - El seguimiento por parte del cliente.
   - Los procedimientos a seguir cuando el cliente quiera cambiar un requisito.

Revisando la propuesta: objetivos

Ejemplo

Un pequeño empresario nos ha encargado un programa para gestionar su carnicería. En este caso,

- *Para comunicarnos con el cliente le llamaremos al teléfono móvil, y solo nos reuniremos si algún imprevisto varían el presupuesto o los plazos en más de un 20 %.*
- *Se debe entregar un fichero* `.exe` *con un programa de instalación sencillo.*
- *El cliente probará todas las funciones durante una semana.*
- *Dado que es un desarrollo de una aplicación pequeña, no habrá seguimiento durante el desarrollo.*
- *Cuando el cliente quiera cambiar un requisito le enviará un correo electrónico al encargado de definir los requisitos para organizar una reunión en la que se estudiará la viabilidad del cambio.*

Revisando la propuesta: objetivos

Discusión

¿Cómo cambian estos factores si el encargo nos lo hace el responsable de las carnicerías de El Corte Inglés?

Revisando la propuesta: objetivos

4. Identificación de riesgos.
   - Los riesgos durante el desarrollo, como carecer de personal en el área de la aplicación, deben ser identificados.
   - Además, se deben proponer soluciones viables si estos riesgos finalmente aparecen.
   - El estudio de los riesgos es un área complicada en sí misma.

Riesgos imprevisibles

- Esta pregunta es una consulta real en `http://stackoverflow.com/`.
- ¿Es posible evitar estos fallos?

We have an employee whose last name is Null. He kills our employee lookup application when his last name is used as the search term (which happens to be quite often now). The error received (thanks Fiddler!) is

```
<soapenv:Fault>
 <faultcode>soapenv:Server.userException</faultcode>
 <faultstring>coldfusion.xml.rpc.CFCInvocationException: [coldfusion.runtime.Mis
```

Cute, huh?

The parameter's type is string.

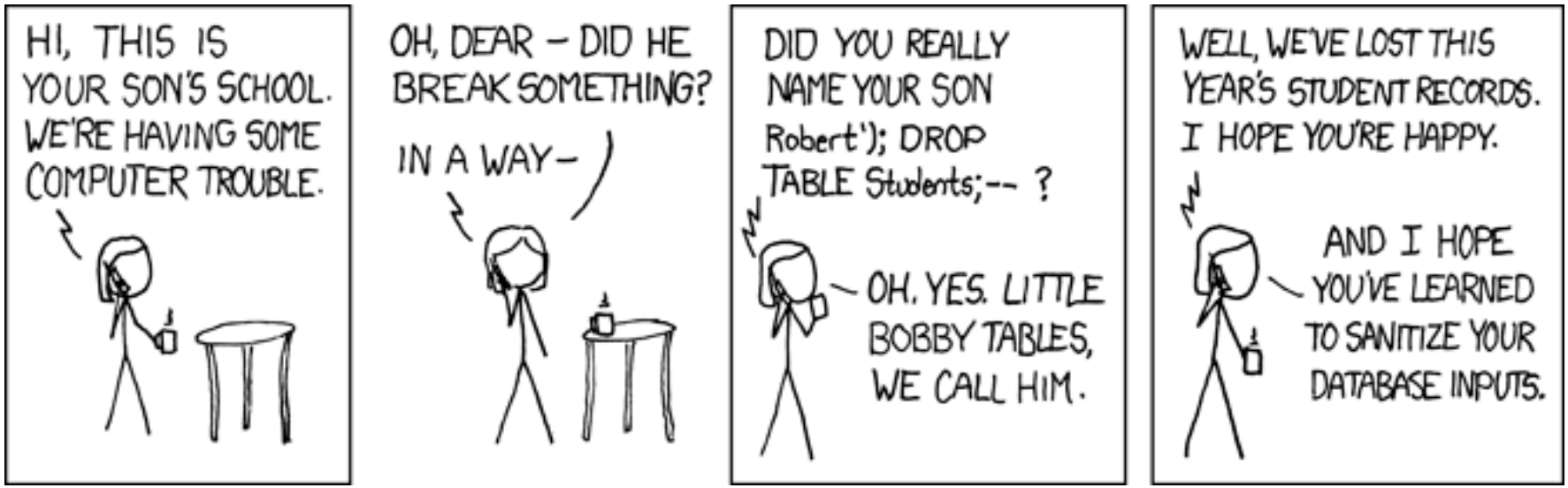
I am using:

- WSDL (SOAP).
- Flex 3.5
- ActionScript 3
- ColdFusion 8

Note that the error DOES NOT occur when calling the webservice as an object from a ColdFusion page.

Riesgos imprevisibles

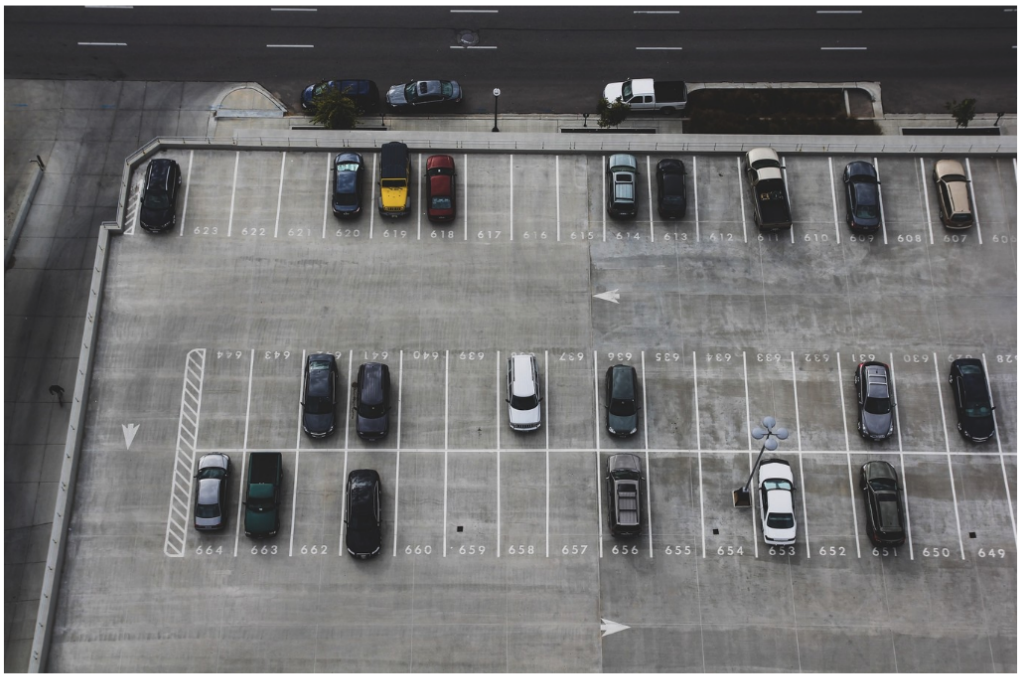
- Una sugerencia similar se hace en esta viñeta de XKCD.



Riesgos imprevisibles

- También en este problema real:

Un informático registró la matrícula 'NULL' para su coche y ahora debe miles de dólares en multas que le llegan por error



13 Agosto 2019 - Actualizado 19 Agosto 2019, 09:28

56 COMENTARIOS

https://www.xataka.com/vehiculos/informatico-registro-matricula-null-para-su-coche-ahora-debe-miles-

Revisando la propuesta: objetivos

4. Identificación de riesgos. Podemos identificar los riesgos como:
   - Riesgos de personal.
   - Riesgos de plazo.
   - Riesgos de funcionamiento.
   - Riesgo de requisitos.
   - Riesgo de subcontrata.
   - Riesgos de uso y rendimiento.

Revisando la propuesta: objetivos

Discusión

- *Un posible riesgo es que el cliente haga muchos cambios en los requisitos. ¿Cómo propondrías solucionar este problema?*
- *Un posible riesgo de personal consiste en la dimisión de un miembro del equipo. Este problema es especialmente crítico en equipos pequeños. ¿Cómo propondrías solucionar este problema?*

Revisando la propuesta: objetivos

5. Estimación adecuada de los recursos del proyecto y de los plazos.
   - Los recursos son tanto el personal como el presupuesto, incluyendo los gastos en subcontratas.
   - La estimación de los plazos debe tener en consideración los requisitos de todas las secciones que participan en el proyecto (e.g. ¿son necesarias interfaces al acabar?).
   - En algunos casos, se hace un presupuesto algo por debajo del coste real para atraer clientes. Una vez se calcula el coste real, esto se considera una pérdida calculada, que se espera cubrir con los beneficios obtenidos en el futuro.

Revisando la propuesta: objetivos

6. Examinar la capacidad de la compañía respecto al proyecto.
   - Este examen debe considerar la disponibilidad de personal especializado y del hardware necesario durante los plazos estimados
7. Examinar la capacidad del cliente para cumplir con los acuerdos.
   - Este examen estudia la capacidad de pago del cliente.
   - También debe estudiar sus infraestructuras y la capacidad del personal para usar el software.

Revisando la propuesta: objetivos

Ejemplo

Se han dado casos de problemas con los presupuestos por no estudiar las capacidades del cliente.
Por ejemplo, si se debe enseñar al personal de todas las tiendas a usar el software y la empresa tiene sedes internacionales el coste se dispara.
Este aspecto está tan ligado al estudio del cliente como al estudio de los riesgos.

Revisando la propuesta: objetivos

8. Definición de la participación de subcontratas y de otras empresas.
   - Este punto se encarga de las garantía de calidad, los plazos de pago, la distribución de beneficios y la cooperación entre todas las partes.
9. Definición y protección de los derechos del software
   - Este aspecto es especialmente importante cuando se reutiliza software o cuando se pretende que el software sea reutilizado en el futuro.
   - También se refiere al uso de ficheros con información personal y a su seguridad.

Revisando el contrato: objetivos

Los objetivos de revisar el borrador del contrato buscan asegurar que las siguientes actividades se han llevado a cabo de forma satisfactoria:

1. No quedan puntos sin clarificar en el borrador.
2. Todas las revisiones hechas entre el cliente y la compañía están documentadas de manera completa y correcta en el contrato y sus apéndices.
3. No hay cambios, omisiones o añadidos que no se hayan discutido y sobre los cuales se haya llegado a un acuerdo que no estén en el contrato. Por tanto, cualquier cambio precisará nuevos compromisos.

Implementación de la revisión del contrato

- La revisión del contrato varía según el proyecto.
- Según el proyecto puede haber mayores dificultades técnicas o de organización.
- Se pueden distinguir distintos niveles según las partes del contrato que se deben revisar.
- Distintas personas pueden (y deberían) revisar distintas partes.

Factores que afectan a la revisión del contrato

- Los principales factores que afectan a la revisión del contrato son:

  La magnitud del proyecto. Se suele medir en personal por mes.
  La complejidad técnica del proyecto.
  La complejidad de organización del proyecto. Depende del número de clientes y de compañías estén involucradas.
  La experiencia del personal en el área. Este factor también tiene en cuenta las posibilidad de reutilización de código.
- Cuanto más complejo sea cada uno de estos factores, más personas serán necesarias para revisar el contrato.

¿Quién se encarga de la revisión del contrato?

- Dependiendo de la complejidad del proyecto, su revisión correrá a cargo de (listado en orden ascendente de complejidad):
  - Un miembro cualquiera del equipo que hizo la propuesta (no necesariamente el jefe).
  - Varios miembros del equipo que hizo la propuesta.
  - Varios miembros del equipo y un profesional externo.
  - Los miembros del equipo y un equipo de expertos.
- Además, se puede contratar a externos cuando los miembros del equipo no tengan experiencia, por pequeño que sea el equipo.

Revisiones para propuestas complejas

- Consideramos que una propuesta es compleja cuando al menos uno de los factores mostrados anteriormente presenta una complejidad alta.
- Las mayores dificultades que suelen presentar las revisiones de estos proyectos, y que por tanto pueden presentar problemas, son:

  Presiones de tiempo. Los clientes de proyectos grandes quieren empezar cuanto antes, por lo que se presiona a todas las partes para firmar rápido el contrato.

  Necesidades de expertos. Cuando se trabaja en un área compleja es necesario dedicar parte del presupuesto a expertos que puedan estudiar la propuesta adecuadamente.

  Los encargados de la revisión están ocupados. En general, los encargados de la revisión son personas con experiencia y, por tanto, con muchas obligaciones, por lo que no pueden dedicar todo su tiempo a una única tarea.

Revisiones para propuestas complejas

- Una vez tenemos claro que hay problemas, podemos proponer soluciones:

  La propia revisión necesita plazos. Idealmente, los plazos de la revisión forman parte de los plazos del proyecto.

  La revisión la lleva a cabo un equipo. De esta manera se pueden dividir y organizar las tareas.

  Es necesario un líder de la revisión. Entre sus tareas estarán elegir el equipo, dividir las tareas, coordinar a los miembros del equipo y coordinarse con el cliente.

Planes de desarrollo y calidad

- Supongamos que nos acaban nombrar líder de un equipo de desarrollo para un proyecto importante.
- Lo más seguro es que nos veamos presionados desde el primer día para cumplir los plazos sin salirnos del presupuesto.
- Aunque te dispongas a seguir las ideas del proyecto, seguramente los plazos y las disponibilidades hayan variado.
- Es mejor revisar los planes que se propusieron en la propuesta.
- Si se siguen ajustando no habrá supuesto mucho trabajo y podemos seguir usándolos.
- En otro caso, prepararlos nos habrá servido para seguir cumpliendo con los objetivos.

Planes de desarrollo y calidad

- Estos nuevos planes deben:
  - Tener en cuenta los cambios habidos desde la propuesta (e.g. personal contratado/despedido, material estropeado, etc).
  - Dar más detalles sobre plazos y presupuestos (al haber avanzado el proyecto se puede hacer una estimación mejor).
  - Añadir nuevos requisitos.
  - Preparar los planes en caso de ocurrir cualquier riesgo.

Objetivos del plan de desarrollo

- Más concretamente, los objetivos del plan de desarrollo son:
  - Poner plazo a las actividades de desarrollo necesarias para terminar a tiempo el proyecto.
  - Decidir los miembros del equipo de desarrollo y asignarles tareas y plazos.
  - Solucionar los riesgos que aparezcan en el desarrollo.
  - Implementar las actividades de garantía de calidad.
  - Generar la documentación asociada a las actividades de control.

Elementos del plan de desarrollo

- Para cumplir estos objetivos, debemos tener en cuenta los siguientes elementos:

1. Los productos del plan, que incluyen:
   - Documentación sobre el diseño, incluyendo fechas de complección e indicando qué se debe entregar al cliente.
   - Documentación sobre los productos software, incluyendo fechas de complección e instrucciones sobre instalación.
   - Actividades de formación, indicando fechas, participantes y lugar.

Elementos del plan de desarrollo

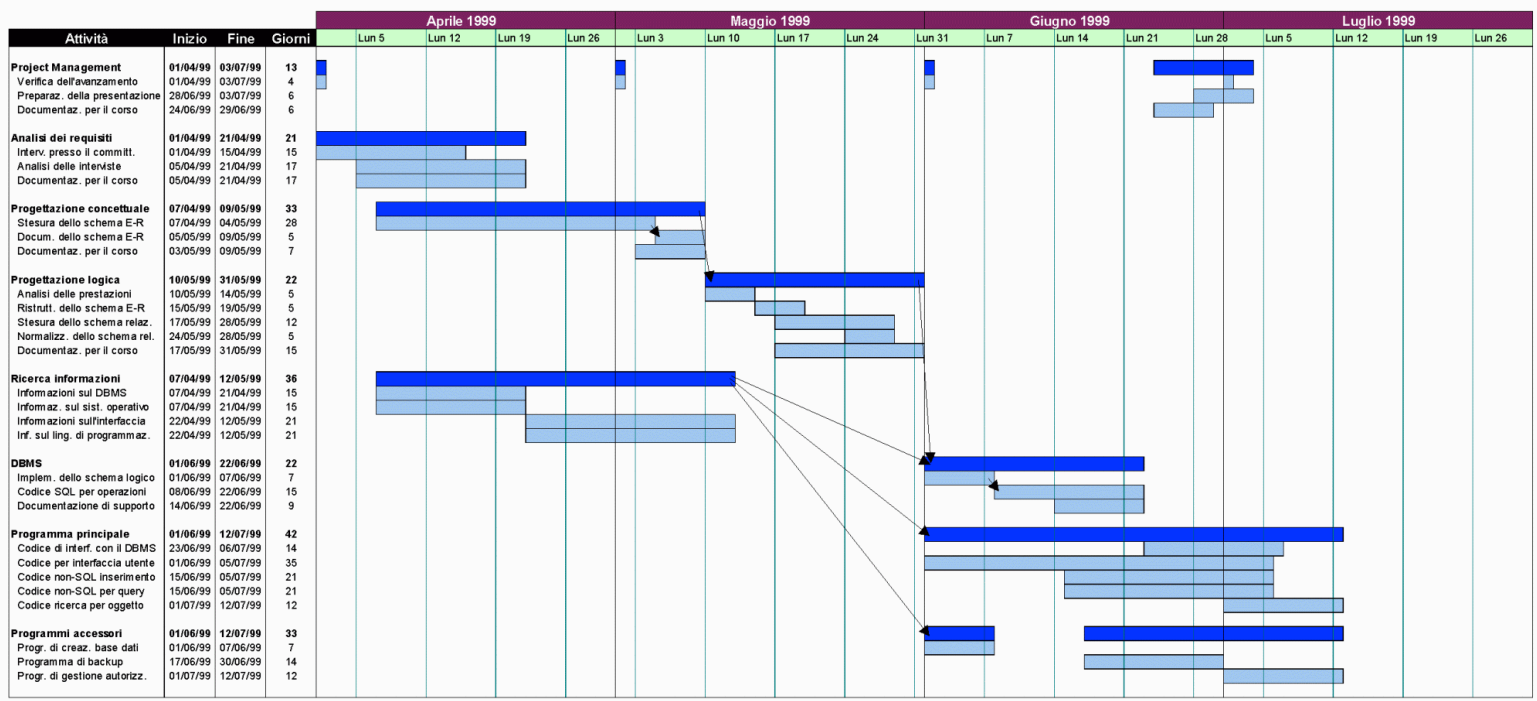
2. Las interfaces, que incluyen
   - Las interfaces software con módulos ya existentes.
   - Las interfaces con el hardware disponible.
   - Las interfaces (tanto software como hardware) con el resto de equipos de desarrollo.
3. Las herramientas y directivas que se deben usar en cada fase del proyecto.
   En este caso se debe ser cuidadoso con las partes del proyecto subcontratadas, pues pueden estar acostumbrados a trabajar con otras metodologías.

Elementos del plan de desarrollo

4. Los estándares y procedimientos software.
5. La descripción del proceso de desarrollo.
   - Se deben dar los suficientes detalles de cada fase del proyecto.
   - Además, se deben dar las actividades (las tareas "atómicas") planeadas:
     - Duración de cada actividad.
     - Las dependencias entre las actividades y por tanto el orden en el que deben realizarse.
     - Los recursos necesarios para cada actividad.
   - Hay muchas herramientas gráficas para esta parte del plan de desarrollo.

Elementos del plan de desarrollo

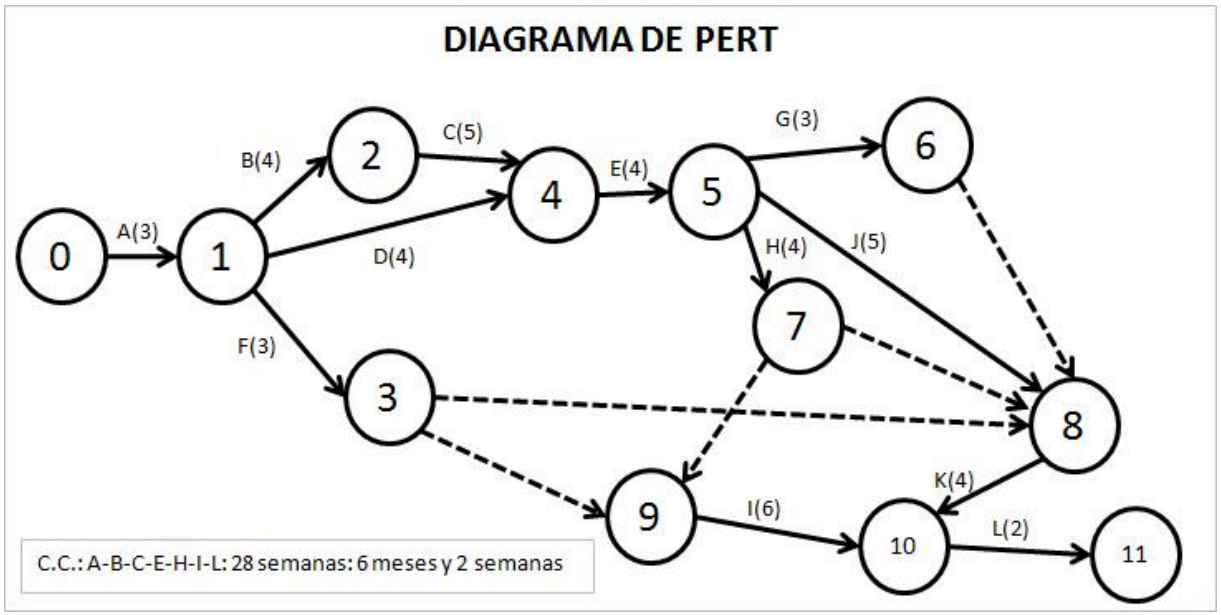
5. La descripción del proceso de desarrollo.
   - Es frecuente usar diagramas Gantt para ver las dependencias y los plazos.



Attività	Inizio	Fine	Giorni
Project Management	**01/04/99**	**03/07/99**	**13**
Verifica dell'avanzamento	01/04/99	03/07/99	4
Preparaz. della presentazione	28/06/99	03/07/99	6
Documentaz. per il corso	24/06/99	29/06/99	6
Analisi dei requisiti	**01/04/99**	**21/04/99**	**21**
Interv. presso il committ.	01/04/99	15/04/99	15
Analisi delle interviste	05/04/99	21/04/99	17
Documentaz. per il corso	05/04/99	21/04/99	17
Progettazione concettuale	**07/04/99**	**09/05/99**	**33**
Stesura dello schema E-R	07/04/99	04/05/99	28
Docum. dello schema E-R	05/05/99	09/05/99	5
Documentaz. per il corso	03/05/99	09/05/99	7
Progettazione logica	**10/05/99**	**31/05/99**	**22**
Analisi delle prestazioni	10/05/99	14/05/99	5
Ristrutt. dello schema E-R	15/05/99	19/05/99	5
Stesura dello schema relaz.	17/05/99	28/05/99	12
Normalizz. dello schema rel.	24/05/99	28/05/99	5
Documentaz. per il corso	17/05/99	31/05/99	15
Ricerca informazioni	**07/04/99**	**12/05/99**	**36**
Informazioni sul DBMS	07/04/99	21/04/99	15
Informaz. sul sist. operativo	07/04/99	21/04/99	15
Informazioni sull'interfaccia	22/04/99	12/05/99	21
Inf. sul ling. di programmaz.	22/04/99	12/05/99	21
DBMS	**01/06/99**	**22/06/99**	**22**
Implem. dello schema logico	01/06/99	07/06/99	7
Codice SQL per operazioni	08/06/99	22/06/99	15
Documentazione di supporto	14/06/99	22/06/99	9
Programma principale	**01/06/99**	**12/07/99**	**42**
Codice di interf. con il DBMS	23/06/99	06/07/99	14
Codice per interfaccia utente	01/06/99	05/07/99	35
Codice non-SQL inserimento	15/06/99	05/07/99	21
Codice non-SQL per query	15/06/99	05/07/99	21
Codice ricerca per oggetto	01/07/99	12/07/99	12
Programmi accessori	**01/06/99**	**12/07/99**	**33**
Progr. di creaz. base dati	01/06/99	07/06/99	7
Programma di backup	17/06/99	30/06/99	14
Progr. di gestione autorizz.	01/07/99	12/07/99	12

Elementos del plan de desarrollo

5. La descripción del proceso de desarrollo.
   - También se usan diagramas PERT para ver las dependencias y el tiempo en el camino más largo.



   - Algunas herramientas, como Microsoft Project, permiten diseñar estos diagramas y realizar el seguimiento de las tareas y los hitos.

Elementos del plan de desarrollo

6. Hitos.
   - Se debe definir su plazo y los resultados necesarios.
7. Organización del personal, que incluye:
   - Definición de los equipos y sus tareas, incluyendo las subcontratas.
   - Los requisitos profesionales, como certificados, experiencia en un lenguaje o herramienta, etc.
   - Personal en cada equipo en cada periodo de tiempo.
   - Elección de los líderes de proyecto. Se encargarán de organizar el equipo y comunicarse con el líder del siguiente nivel y con los líderes del resto de equipos.

Elementos del plan de desarrollo

8. Infraestructuras, que incluyen:
   - Software y hardware.
   - Despachos.
   - Cualquier otro material adicional.
   - Para cada recurso se deben indicar los plazos en los que son necesarios.

Ejemplo

El software para automóviles se prueba con coches reales al final del proceso de desarrollo.

Elementos del plan de desarrollo

9. Riesgos.
   - Un riesgo es "un estado o propiedad de una tarea que, si se ignora, incrementa las posibilidades de fracaso del proyecto".
   - Los riesgos más típicos son:
     - Carencias en la preparación del personal.
     - Falta de personal.
     - Problemas con las subcontratas.
   - El modelo en espiral de desarrollo tiene una fase de control de riesgos precisamente porque es un problema que aparece frecuentemente.

El *top ten* de los riesgos

Discusión

¿Cuáles creéis que son los 10 *principales riesgos que aparecen en el desarrollo de software?*

El *top ten* de los riesgos

1. Riesgos de personal.
2. Riesgos de plazos.
3. Riesgos de funcionalidad (no programar lo solicitado).
4. Riesgos de interfaz.
5. Riesgos de programar requisitos innecesarios.
6. Riesgos de (demasiadas) modificaciones de los requisitos por parte del cliente.
7. Riesgos de baja calidad de los productos subcontratados.
8. Riesgos de baja calidad en las tareas (no software) subcontratadas.
9. Riesgo de baja productividad.
10. Riesgos "científicos" (los requisitos son demasiado complejos/no computables).

Elementos del plan de desarrollo

10. Métodos de control.
    - Se debe indicar cómo se va a seguir el progreso y cómo y cuándo se van a producir las reuniones de coordinación.
11. Estimación de costes.
    - La estimación de costes debe tener todos los factores en cuenta: personal, infraestructuras, riesgos, etc.
    - Se debe tener especialmente en cuenta los costes por subcontratación.
    - Los cambios de personal suelen variar mucho el presupuesto.

Elementos del plan de desarrollo

Discusión

¿Cuáles serían los elementos del plan de desarrollo para una aplicación de venta de libros, que permite venta online y venta en la propia tienda?

Elementos del plan de calidad

- También debemos distinguir ciertos elementos en el plan de calidad.

1. Objetivos de calidad.
   - Se refieren a los objetivos que debe cumplir el producto entregado.
   - En general son mejores las medidas cuantitativas que las cualitativas.
   - Estas medidas dan en general datos más objetivos del rendimiento durante las pruebas.
   - Obviamente, ambas medidas están relacionadas.

Elementos del plan de calidad

Ejemplo (Asistencia telefónica)

Nos han encargado hacer un programa para ayudar a los encargados de la asistencia telefónica de una cierta compañía de telefonía móvil.
El cliente pide que sea "sencilla de usar", "eficiente" y "que ofrezca un buen servicio" (que son medidas cualitativas) pero, dado que estas palabras pueden tener un significado diferente para el cliente y para nosotros, le pedimos más detalles.
¿Cómo definiríais vosotros estas cualidades?
Una posible correspondencia entre estas medidas y las correspondientes medidas cualitativas se muestra a continuación.

Elementos del plan de calidad

Ejemplo (Asistencia telefónica)

Medidas cualitativas	*Medidas cuantitativas*
Interfaz amigable	*Un nuevo trabajador tiene que aprender a usarlo en menos de* 8 *horas, y a configurarlo en menos de* 5 *días de trabajo.*
Confiable	*No puede sumar fallos por más de* 30 *minutos a la semana.*
Debe funcionar continuamente	*En caso de error se debe recuperar en menos de* 10 *minutos.*

Elementos del plan de calidad

Ejemplo (Asistencia telefónica)

Medidas cualitativas	*Medidas cuantitativas*
Eficiente	*Tiene que permitir atender al menos a 100 clientes en una jornada de 8 horas.*
Confiable	*No puede sumar fallos por más de 30 minutos a la semana.*
Debe dar gran calidad a sus clientes	*Un cliente no debe esperar más de 30 segundos por cada llamada que realice debido a retrasos en el software.*

Elementos del plan de calidad

2. Actividades de revisión planificadas
   - El plan de calidad debe incluir una lista completa de las actividades de revisión que se planea realizar.
   - Para cada actividad (inspección del diseño, inspección del código, etc.) se debe indicar:
     - Qué se quiere inspeccionar.
     - El plazo en el que se quiere realizar.
     - Los procedimientos a aplicar.
     - El encargado de la revisión.

Elementos del plan de calidad

3. Pruebas del software planificadas
   - El plan de calidad debe incluir una lista completa de las actividades de testing que se vayan a realizar.
   - Para cada test se debe indicar:
     - El módulo al que se va a aplicar.
     - El tipo de testing que se va a aplicar.
     - El plazo en el que se quiere realizar.
     - Los procedimientos a aplicar.
     - El encargado de la revisión.

Elementos del plan de calidad

4. Pruebas necesarias para aceptar software externo
   - El plan de calidad debe indicar qué pruebas debe pasar el software externo para garantizar su funcionamiento.
   - Consideramos como software externo:
     - El software comprado.
     - El software proporcionado por el cliente.
     - El software producido por subcontratas.

Discusión

¿Qué software consideras que planteará más problemas? ¿Y menos?

- Control de configuraciones
  - El plan de calidad debe especificar el software que se usará para hacer el seguimiento del proyecto y de los cambios que puedan producirse.

Elementos del plan de calidad

Discusión

¿Cómo sería el plan de calidad de nuestra aplicación para venta de libros?

Discusión

¿Tiene sentido tanto plan para una aplicación tan pequeña?

Planes de desarrollo y calidad en proyectos pequeños y en proyectos internos

- Las características de las que hemos hablado hasta ahora son para proyectos grandes.
- Aunque pueden aplicarse a proyectos pequeños, no tendría sentido en general, puesto que los plazos, el presupuesto y el personal disponible será menor.
- Complicar mucho un proceso que debería ser sencillo puede afectar negativamente al cliente.
- Si el proyecto es interno se comparten ciertos conocimientos y se respetan ciertos acuerdos implícitos, que por tanto no hace falta revisar.
- Es posible que haya que estudiar algunos elementos que no aparecían en proyectos grandes.

Planes de desarrollo y calidad en proyectos pequeños

- Podemos distinguir distintos tipos de proyectos "pequeños":
  - Aquellos que no necesitan planes de desarrollo ni de calidad, en general los que duran menos de 15 días.
  - Aquellos en los que la necesidad de planes de desarrollo y de calidad depende de la decisión del líder de proyecto.
  - En general estos proyectos necesitan alrededor de 2 meses de trabajo, y para ellos no se detectan riesgos importantes.
  - Aquellos en que los planes de desarrollo y de calidad son obligatorios.

Planes de desarrollo y calidad en proyectos pequeños

- En caso de ser necesarios los planes de desarrollo y calidad, puede no ser necesario incluir todos los elementos.
- Para el plan de desarrollo se recomienda tener:
  - Los productos que deben entregarse.
  - Las especificaciones que deben superar los productos.
  - Los riesgos del desarrollo.
  - La estimación de costes.
- Para el plan de calidad se recomienda tener:
  - Los objetivos de calidad.

Planes de desarrollo y calidad en proyectos pequeños

Discusión

¿Qué ventajas crees que se obtienen elaborando los planes de desarrollo y calidad en proyectos pequeños? ¿Qué desventajas encuentras?

Ventajas de elaborar planes de desarrollo y calidad en proyectos pequeños

- Se tienen más claros los objetivos.
- Se pueden asignar las tareas con más claridad.
- Es más fácil hacer el seguimiento, tanto para el cliente como para la compañía.
- Es más fácil detectar los problemas.
- Los plazos se pueden seguir con claridad.

Planes de desarrollo y calidad en proyectos internos

- Un proyecto interno es aquel que se desarrolla para uso de otros departamentos dentro de una misma empresa.
- La idea principal es que no hay un "cliente".
- Estos proyectos pueden ser tanto grandes como pequeños.
- Debe ser la importancia del proyecto, y no el hecho de que sea interno, lo que determine si se desarrollan planes de desarrollo y calidad.

Contenidos

Actividades de calidad durante el ciclo de vida del software

- El ciclo de vida del software consta de dos partes:
  - La fase de desarrollo.
  - La fase de mantenimiento.
- Durante la fase de desarrollo se deben detectar errores de diseño y de implementación mediante:
  - Revisiones formales del diseño.
  - Revisión por pares.
  - Opiniones de expertos.
  - Testing.
- Durante el mantenimiento, debemos:
  - Corregir.
  - Adaptar.
  - Mejorar.

Contenidos

Actividades de calidad durante el desarrollo

- Para explicar las actividades de calidad durante el desarrollo empezaremos recordando los distintos modelos de desarrollos que se usan en la actualidad.
- Después veremos:
  - Los objetivos de la garantía de calidad en el desarrollo.
  - Qué elementos debemos considerar antes de implementar la garantía de calidad.
  - Cómo integrar la garantía de calidad en el desarrollo.

Los modelos de desarrollo de la Ingeniería del software

Discusión

¿Qué modelos habéis estudiado/usado?

Los modelos de desarrollo de la Ingeniería del software

El modelo en cascada. Es el modelo más antiguo, que distingue las distintas fases y las aplica secuencialmente.

El modelo de prototipado. Consiste básicamente en crear un prototipo básico e irle añadiendo funcionalidad a medida que el cliente aprueba cada fase.

El modelo en espiral. Es un modelo iterativo que incluye revisiones y cambios por parte del cliente, estudio de riesgos, diseño e ingeniería.

El modelo orientado a objetos. Tiene especialmente en cuenta la reutilización de código.

Modelos ágiles. Iterativos e incrementales, se organizan en grupos auto-organizados y manejan bien el cambio.

El modelo en cascada

- Este modelo consta de 7 fases que se aplican secuencialmente:
  1. Definición de requisitos.
  2. Análisis de requisitos.
  3. Diseño.
  4. Implementación.
  5. Testing.
  6. Instalación.
  7. Mantenimiento.

El modelo en cascada

- Según la complejidad del proyecto, alguna fase puede a su vez dividirse en subfases.
- Al acabar cada fase, se estudian los resultados (tanto los desarrolladores como el cliente).
- Los resultados del estudio son:
  - Aprobación de los resultados y paso a la siguiente fase.
  - Solicitudes de corregir, rehacer o cambiar partes.
- En ciertos casos los cambios pueden llevar a retroceder varias fases.
- En tal caso se usa el llamado modelo con retroalimentación, que permite volver a fases anteriores y continuar el desarrollo a partir de ellas.

El modelo de prototipado

- En este modelo, el cliente da su opinión sobre los distintos modelos que le van siendo presentados.
- Para aplicar el modelo de prototipado es necesario:
  1. Trabajar en un área en la que sea posible desarrollar modelos del sistema de manera rápida.
  2. El cliente esté dispuesto a participar de forma más activa en el proceso de desarrollo.
- Básicamente, podemos considerar que tenemos 3 fases consecutivas:
  1. Diseño del prototipo.
  2. Implementación del prototipo.
  3. Evaluación por parte del cliente.
- Cuando la evaluación es negativa se cambia el diseño teniendo en cuenta las opiniones del cliente.
- Cuando la evaluación es positiva se pasa a los tests y, tras ellos, a la instalación y mantenimiento.

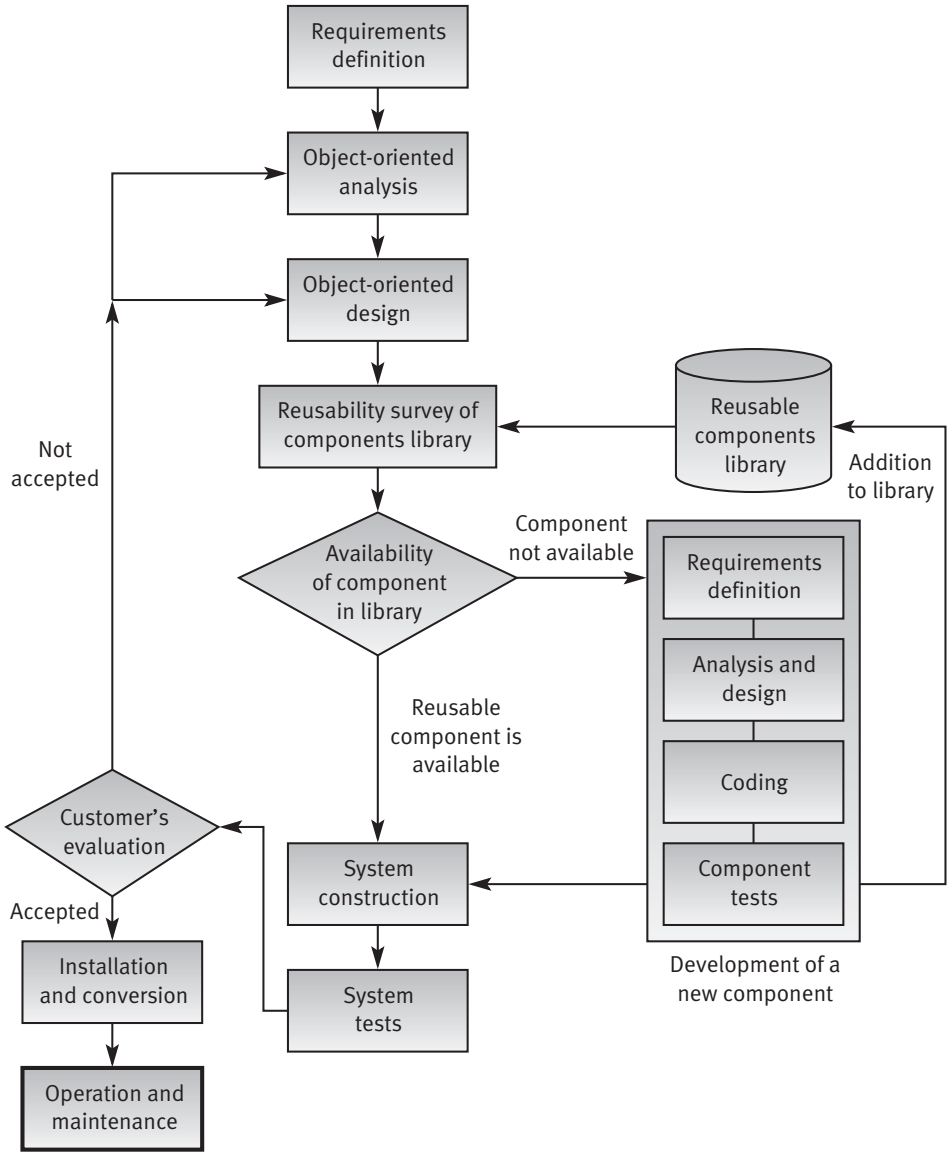
El modelo en espiral

- El modelo en espiral está pensado para desarrollar proyectos complejos y largos.
- Es un proceso iterativo que añade, con respecto a los modelos anteriores, evaluación de riesgos.
- Sus fases son:
  1. Planificación.
  2. Análisis y resolución de riesgos.
  3. Actividades de ingeniería: diseño, implementación, testing, instalación y lanzamiento (según la iteración).
  4. Evaluación por parte del cliente: comentarios, cambios, requisitos adicionales, etc.

El modelo orientado a objetos

- En primer lugar, hay que tener claro que los otros modelos también sirven para programar con orientación a objetos.
- Este modelo se caracteriza por el énfasis en la reutilización de código.
- Por tanto, en la primera fase del modelo se realiza un estudio del proyecto.
- Cuando se desarrolla nuevo software se almacena para futura reutilización.

El modelo orientado a objetos



Fuente: Daniel Galin, Software quality assurance: from theory to implementation, Addison-Wesley, 2004.

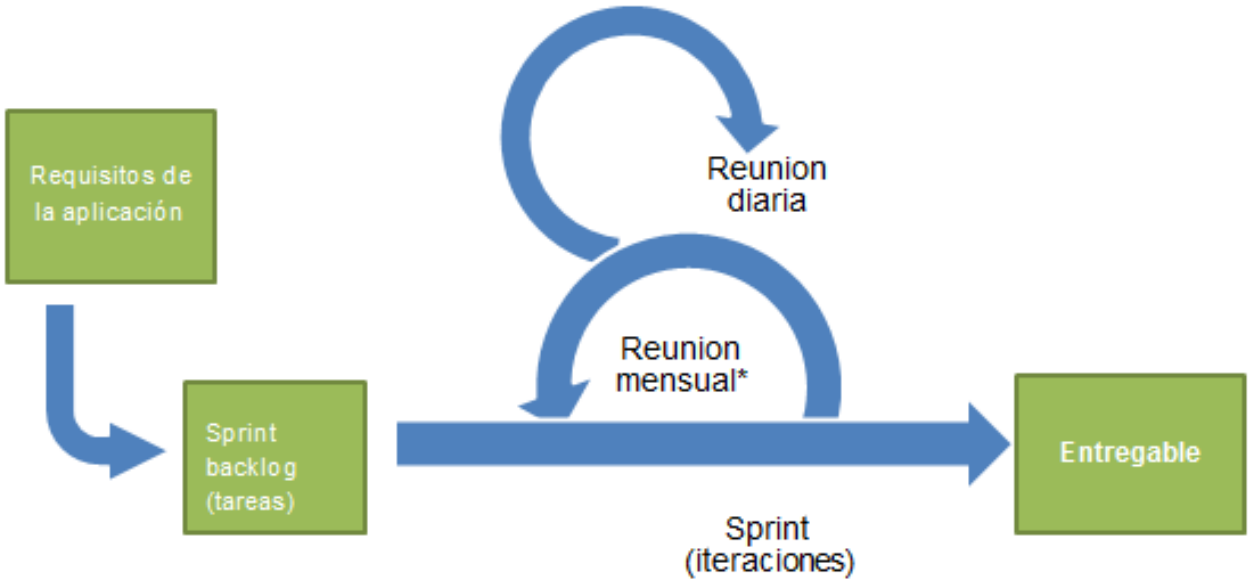
Scrum

- Roles:
  - Scrum Master, que "facilita" el proceso y la gestión de cambios.
  - Dueño del producto, que prioriza objetivos.
  - Interesados (*stakeholders*), son los clientes, usuarios, etc.
  - Equipo, a cargo del desarrollo.
- Sprints:
  - Periodos entre una y cuatro semanas (lo más cortos posible).
  - Se crea funcionalidad entregable.

Scrum

- Adaptación continua:
  - Gestión de las expectativas del cliente.
  - Control de riesgos.
  - Adaptación a cambios.
  - Gestión del grupo.
  - Etcétera.
- Equipos auto-dirigidos y auto-organizados.
- Reunión Scrum diaria, en la que se presenta el avance del día anterior y los objetivos del día.

SCRUM



Fuente: `https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)#/media/File:Scrumm.png`

Objetivos de calidad durante el desarrollo

- Se debe fijar un objetivo de calidad a cada fase del proyecto de desarrollo.
- Además, cada hito o fecha clave debe tener también ciertos objetivos de calidad asociados.
- Los encargados de la calidad deben definir la lista de actividades de calidad.
- Para cada actividad, es necesario fijar:
  - El periodo en el que se llevará a cabo.
  - Cómo se debe llevar a cabo.
  - El personal y los recursos necesarios.
  - Qué hacer si supone modificaciones en los requisitos.

Objetivos de calidad durante el desarrollo

- Las actividades de calidad dependen del proyecto y del personal.
- Según el proyecto:
  - Tamaño del proyecto.
  - Complejidad técnica del proyecto.
  - Posibilidades de reutilización de código.
  - Perjuicios en caso de errores.
- Según el personal:
  - Cualificación del personal.
  - Experiencia del personal en el equipo y el área.
  - Disponibilidad para contratar expertos.

Objetivos de calidad durante el desarrollo

Ejemplo (Tienda informática)

Nos han encargado un programa para llevar el inventario (incluyendo ventas, devoluciones, etc) de una tienda de informática. Durante los 2 últimos años hemos recibido muchas solicitudos de este tipo; de hecho, este es el 15° encargo de este tipo que tenemos.
¿Cómo creéis que deben ser las actividades de calidad? ¿Qué modelo de desarrollo usarías?

Objetivos de calidad durante el desarrollo

Ejemplo (Tienda informática)

En principio nos basta con unas pocas actividades de calidad, ya que la aplicación es sencilla, el equipo la conoce bien y podemos reutilizar buena parte del código. Podemos considerar que bastaría con:

- *Revisión de los requisitos, que se llevará a cabo al acabar el análisis, durará medio día y puede suponer, como máximo, cambios que nos lleven un día.*
- *Inspección del diseño, que se llevará a cabo al acabar el diseño, puede durar un día y suponer cambios que nos lleven otro día.*
- *Testing, que se llevará a cabo al acabar la implementación. Durará tres días y nos puede suponer dos días de modificaciones.*

Objetivos de calidad durante el desarrollo

Ejemplo (Hospital)

Nos han encargado el software para controlar el estado de los pacientes en un hospital usando las nuevas máquinas que han recibido. Es necesario que las nuevas máquinas se conecten al antiguo sistema de monitorización, y además que comunique novedades a la sala de enfermería.
¿Cómo creéis que deben ser las actividades de calidad? ¿Qué modelo de desarrollo usarías?

Objetivos de calidad durante el desarrollo

Ejemplo (Hospital)

En este caso la aplicación es mucho más compleja, como las máquinas son nuevas no tenemos experiencia ni posibilidad de reutilizar código, y es posible que no tengamos ningún experto. Además, un fallo en el sistema puede suponer graves consecuencias.
En principio, sería necesario hacer comprobaciones de calidad al acabar cada fase. Estas actividades llevarán varios días en todos los casos, y causan también cambios de varios días.

Verificación, validación y cualificación

- Hay tres conceptos de calidad especialmente importantes durante el desarrollo: la verificación, la validación y la cualificación.
- Según el IEEE
  - "Verificación es el proceso de evaluar un sistema o un componente para determinar si los productos generados por una cierta fase del desarrollo satisfacen las condiciones impuestas al comienzo de la fase".
  - "Validación es el proceso de evaluar un sistema o un componente durante o al final del proceso de desarrollo para determinar si cumple los requisitos".
  - "Cualificación es el proceso para determinar si un sistema o componente es adecuado para ser usado".

Verificación, validación y cualificación

- Es decir, la verificación comprueba que el último paso que hemos dado en el desarrollo se ha hecho como se esperaba.
- La validación comprueba que seguimos los planes que teníamos al principio del desarrollo.
- El cualificación comprueba que funciona.

Un modelo para calcular la efectividad y el coste de eliminar defectos

- Para acabar, veremos un modelo cuantitativo concreto para eliminar defectos.
- Este modelo estudia:
  - La efectividad de las actividades para eliminar defectos.
  - El coste de eliminar los defectos.
- En primer lugar veremos las ideas básicas sobre efectividad y coste.
- Después veremos los detalles del modelo.

Un modelo para calcular la efectividad y el coste de eliminar defectos: Origen de los defectos

- La distribución del origen de los defectos indica en qué fase del proceso de desarrollo se introducen los defectos.
- En general, los errores se distribuyen como sigue:

Fase	Defectos introducidos
Especificación de requisitos	15 %
Diseño	35 %
Implementación	40 %
Documentación	10 %

Un modelo para calcular la efectividad y el coste de eliminar defectos: Efectividad

- Lo normal es que cualquier actividad de calidad, por básica que sea, pueda eliminar un cierto porcentaje de defectos.
- Hay que tener en cuenta que no basta con encontrar los defectos, también es necesario arreglarlos adecuadamente.
- Se estima que alrededor del 10 % de los defectos no se arregla adecuadamente, por lo que siguen siendo defectos (peores que antes, porque se cree que están solucionados).
- Los errores que no se eliminan van pasando por las distintas fases del desarrollo.
- Cada fase de calidad se enfrenta a defectos "mezclados", causados por los defectos no corregidos en fases anteriores.
- En general se espera una fase de calidad elimine al menos el 40 % de los defectos.

Un modelo para calcular la efectividad y el coste de eliminar defectos: Coste

- El coste de eliminar el coste varía según la fase, pero en general es más costoso cuanto más avanzado está el desarrollo.
- Por ejemplo, si eliminar un defecto de diseño en la propia fase de diseño cuesta 2 días y medio, eliminar ese mismo error en la fase de testing puede suponer 40 días.
- A pesar de la introducción de técnicas de garantía e ingeniería del software, los costes no han mejorado mucho en las últimas décadas (posiblemente porque queden solo a un nivel teórico).

Un modelo para calcular la efectividad y el coste de eliminar defectos: El modelo

- El modelo supone que:
  - El desarrollo es lineal y secuencial (como el modelo en cascada).
  - Se introducen defectos en cada fase.
  - Las revisiones y el testing sirven de filtros para eliminar defectos.
  - En cada fase, el total de defectos es la suma de los no arreglados y los añadidos en esta fase.
  - El coste de eliminar los defectos se calcula multiplicando el número de defectos por el coste de eliminar un defecto en esta fase.
  - El resto de defectos pasan al cliente y acabará detectándolos con el uso. El coste en este caso es máximo.

Un modelo para calcular la efectividad y el coste de eliminar defectos: El modelo

- El porcentaje promedio de defectos eliminados es como sigue:

Fase	Eficiencia
Revisión de requisitos	50 %
Revisión del diseño	55 %
Revisión del código	65 %
Pruebas de unidad	50 %
Pruebas de unidad después de revisar el código	30 %
Pruebas de integración	50 %
Pruebas de sistema/aceptación	50 %
Pruebas de documentación	50 %

Un modelo para calcular la efectividad y el coste de eliminar defectos: El modelo

- El coste de eliminar un defecto es como sigue:

Fase	Coste
Implementación	1
Diseño	2,5
Revisión del código	6
Pruebas de unidad	5
Pruebas de unidad después de revisar el código	30

- En este caso el "coste" se representa por unidades y se muestra la proporción entre fases.
- En un proyecto concreto, una unidad de coste puede ser, por ejemplo, "3 días de trabajo" o "500 euros".

Un modelo para calcular la efectividad y el coste de eliminar defectos: El modelo

- Dados los siguientes valores:
  - Defectos Originados por Fase (DOF) (se suele seguir la distribución de la tabla).
  - Defectos Pasados de la fase anterior (DP).
  - Porcentaje de Efectividad de los filtros (% EF) (mirar tabla).
  - Defectos Eliminados en esta fase (DE).
  - Coste de Eliminar un Defecto (CED) (mirar tabla).
  - Coste Total de eliminación de defectos en la fase (CT): DE $\times$ CED.

Un modelo para calcular la efectividad y el coste de eliminar defectos: El modelo

Ejemplo

Supongamos que estamos en la fase de especificación de requisitos, y que tenemos 15 *fallos (DOF).*

- *Como estamos en la primera fase, DP es* 0*.*
- *Según la tabla, % EF es del* 50 %*.*
- *Por lo tanto arreglamos la mitad de los defectos,* 7,5 *(aunque no tenga sentido arreglar "medio defecto", esto es una estimación del coste).*
- *Según la tabla de costes, arreglar un defecto en esta fase (CED) es de* 1 *unidad, por lo que el coste total de eliminación de esta fase es de* 7,5 *unidades de coste.*

Un modelo para calcular la efectividad y el coste de eliminar defectos: El modelo

Ejemplo

Pasamos a la fase de diseño.

- *Si en la especificación de requisitos se introdujeron* 15 *defectos, en esta habrá aproximadamente* 35*.*
- *De la fase anterior han pasado* 7,5 *defectos, por lo que en total tenemos* 42,5*.*
- *Según la tabla, % EF es del* 55 %*.*
- *Por lo tanto arreglamos, redondeando al alza,* 23,5 *defectos.*
- *Como arreglar un defecto en esta fase tiene un coste de* 2,5*, el coste de solucionar los defectos en esta fase es* 58,75*.*

Un modelo para calcular la efectividad y el coste de eliminar defectos: El modelo

Ejemplo

Supongamos, para simplificar, que la siguiente fase es codificar. En esta fase tenemos:

- *Si en la especificación de requisitos se introdujeron* 15 *defectos, en esta habrá aproximadamente* 40*.*
- *De la fase anterior han pasado* 19 *defectos, por lo que en total tenemos* 59*.*
- *Según la tabla, % EF es del* 65 %*.*
- *Por lo tanto arreglamos, redondeando al alza,* 38,5 *defectos.*
- *Como arreglar un defecto en esta fase tiene un coste de* 6*, el coste de solucionar los defectos en esta fase es* 231 *unidades.*

Un modelo para calcular la efectividad y el coste de eliminar defectos: El modelo

Ejemplo

- *Entre las* 3 *fases que hemos analizado, tenemos un coste de* 297,25 *unidades.*
- *Suponiendo que una "unidad de coste" son* 100 *euros, suponen unos gastos de casi* 30000 *euros.*
- *Aun así, lo más grave es que aún tenemos unos* 20 *defectos, que pueden acabar siendo descubiertos por el usuario.*
- *Esto supondrá más gasto, además de la lógica pérdida de confianza.*

¿Qué estamos viendo?

- Revisemos qué hemos visto hasta hora en esta parte del tema.
- El ciclo de vida del software consta de dos partes:
  - La fase de desarrollo.
  - La fase de mantenimiento.
- Dentro de la fase de desarrollo, en la que estamos, tenemos:
  - Revisiones formales del diseño.
  - Revisión por pares.
  - Opiniones de expertos.
  - Testing.
- Hemos visto primero cómo es el proceso de desarrollo y el coste de la eliminación de errores.
- Ahora veremos las revisiones y las opiniones de los expertos.

Revisiones

- Durante las primeras fases del proyecto se realiza, como hemos visto, documentación que indica las propiedades que debe cumplir el producto.
- Durante las revisiones se comprueba que estas propiedades se van cumpliendo.
- Sin embargo, es a veces complicado que el propio diseñador/programador sea capaz de vez sus propios fallos (quizás porque son errores causados por una interpretación errónea de los requisitos).
- Por ello, es normal que otras personas (externos, expertos, pares, etc) también deban revisar el proyecto.

El objetivo de las revisiones

- Los objetivos directos de las revisiones son:
  - Detectar los errores de análisis y diseño, así como localizar qué partes del proyecto se han visto afectadas por estos cambios.
  - Identificar nuevos riesgos.
  - Localizar las prácticas que se alejan del estilo fijado. Corregir este punto permite mejorar la comunicación y coordinación entre equipos.
  - Aprobar el análisis y el diseño del producto. Esta aprobación permite pasar a la siguiente fase del producto.

El objetivo de las revisiones

- Los objetivos indirectos de las revisiones son:
  - Dar lugar a reuniones informales donde se intercambian las opiniones y conocimientos técnicos sobre las distintas unidades del proyecto.
  - Mantener una base de datos de errores de análisis y diseño que permita evitar estos errores en el futuro.
  - De esta base de datos se espera que mejore la calidad y la eficacia, entre otras características, de futuros productos.

Los distintos tipos de revisiones

- A lo largo del tema veremos distintos tipos de revisiones, según su alcance y el personal dedicado a ello.
- En todas ellas cada participante tiene un rol asignado.
- Además es necesario un coordinador que reparta y centralice el trabajo.
- El resultado de las revisiones son listas de errores con detalles sobre su localización y sus efectos, que deben permitir luego comprobar que se han solucionado.
- Como punto final, el hecho de saber que el trabajo será más tarde revisado por gente externa al equipo suele motivar a los trabajadores a esforzarse más.

Revisiones Formales de Diseño

- El aspecto más particular de las revisiones formales de diseño es que es necesario que sean aprobadas por el diseñador del producto.
- Sin dicha aprobación no se puede pasar a la siguiente fase del desarrollo.
- Estas revisiones se llevan a cabo en cualquier momento en el que se complete un documento de análisis o diseño.
- Esto incluye tanto la documentación de requisitos como la documentación de instalación.

Revisiones Formales de Diseño: Participantes

- Distinguimos entre los participantes al líder de la revisión y al equipo a cargo de la revisión.
- Del líder de la revisión se espera:
  - Conocimientos y experiencia en el área en el que se desarrolla el proyecto.
  - Una antigüedad similar a la del líder del proyecto.
  - Una buena relación con los desarrolladores del proyecto.
  - Ser externo al proyecto.
- Por tanto, el líder de la revisión podría ser el líder de otro proyecto, el jefe del equipo de calidad, el ingeniero jefe o incluso el ingeniero jefe del cliente, en caso de tenerlo (por ejemplo, cuando subcontratamos algo pondremos nuestro propio personal para controlar la calidad).

Revisiones Formales de Diseño: Participantes

- Los miembros del equipo se seleccionan entre el personal con más experiencia, los representantes del cliente y los expertos en el área.
- Es deseable que sean externos.
- Hay que tener cuidado con el tamaño del equipo.
- En general, basta con tener un equipo de entre 3 y 5 personas.
- Demasiadas personas suelen generar problemas de coordinación, y al final empeoran la revisión.

Revisiones Formales de Diseño: Participantes

Discusión

¿Qué creéis que se debe hacer cuando la empresa es pequeña? ¿Qué desventajas hay con estas soluciones? ¿Qué más problemas aparecen en equipos demasiado grandes?

Revisiones Formales de Diseño: Preparación

- La reunión se prepara de diferente manera dependiendo de la posición.
- El líder del equipo debe:
  - Citar a los miembros del equipo.
  - Planificar la reunión.
  - Distribuir los documentos de diseño entre los miembros.
- Es importante planificar las reuniones lo más cerca posible del final de la fase de diseño.
- De esta manera se minimiza la espera para el comienzo de la siguiente fase del desarrollo (o del inicio de las modificaciones, de ser necesario).

Revisiones Formales de Diseño: Preparación

- Por su parte, los miembros del equipo deben:
  - Revisar el documento de diseño y elaborar una lista con todas las sugerencias de modificación.
  - Si los documentos son muy extensos el líder del equipo puede dividirlos, asignando distintas partes a cada miembros.
  - Es importante tener una lista de características que deben ser comprobadas.
  - De esta manera los miembros del equipo se centran en las partes del diseño más importantes desde el punto de vista de la calidad.

Revisiones Formales de Diseño: Preparación

Goldenbug Ltd

Checklist for requirement specification report

Project name: _____

The reviewed document: _____ Version: _____

Item no.	Subject	Yes	No	N.A.*	Comments
1	**The document**				
1.1	Prepared according to configuration management requirements				
1.2	Structure conforms to the relevant template				
1.3	Reviewed document is complete				
1.4	Proper references to former documents, standards, etc.				
2	**Specifying the requirements**				
2.1	Required functions were properly defined and clearly and fully phrased				
2.2	Designed inputs conform with required outputs				
2.3	Software requirement specifications conform with product requirements				
2.4	Required interfaces with external software packages and computerized equipment are fully defined and clearly phrased				
2.5	GUI interfaces are fully defined and clearly phrased				
2.6	Performance requirements – response time, input flow capacity, storage capacity – are correctly defined and fully and clearly phrased				
2.7	All error situations and required system reactions are correctly defined and fully and clearly phrased				
2.8	Data interfaces with other existing or planned software package or products components are correctly defined and fully and clearly phrased				
2.9	Procedures to test fulfillment of the specified requirement are correctly and fully defined and clearly phrased				
3	**Project feasibility**				
3.1	Are the specified requirements feasible considering the project's resources, budget and timetable?				
3.2	Are the specified performance requirements (see 2.6) feasible considering the constraints imposed by other system components and by external systems interfaced with the system?				

Revisiones Formales de Diseño: Preparación

- Los miembros del equipo de desarrollo también tienen ciertas responsabilidades de cara a las revisiones formales de diseño.
- Se espera de ellos que expongan una breve presentación de la estructura del documento de diseño.
- Esta presentación tiene lugar una vez los miembros del equipo de revisión ya han leído el documento y tienen cierta familiaridad con él.
- Por tanto, la presentación se centra en los principales aspectos técnicos que hayan generado más problemas o que necesiten aprobación por cualquier otro motivo.

Revisiones Formales de Diseño: La reunión

- La experiencia del líder para coordinar a los miembros del equipo y ajustarse a los objetivos de la revisión es determinante para el buen funcionamiento de la reunión.
- Aunque los puntos se deben decidir para cada reunión concreta, los puntos a tratar suelen ser:
  1. Un breve resumen del documento de diseño.
  2. Los comentarios de los miembros del equipo.
  3. Discusión para decidir cómo afecta al proyecto cada uno de los comentarios en el punto anterior.

Revisiones Formales de Diseño: La reunión

4. La decisión sobre el documento del diseño, que determina el progreso del proyecto. La decisión puede ser:

   Completamente aprobado. Permite pasar directamente a la siguiente fase del proyecto, aunque puede incluir cambios menores.

   Parcialmente aprobado. Permite pasar a la siguiente fase del proyecto a algunos módulos, mientras que se deben corregir aspectos importantes en el resto.

   Denegado. Se debe repetir el diseño (siguiendo las indicaciones). Esta situación se da cuando muchos módulos deben ser modificados o cuando hay fallos críticos. El nuevo diseño necesitará de nuevo una revisión.

Revisiones Formales de Diseño: El informe

- Es obligación del líder escribir un informe inmediatamente después de la revisión.
- Distribuir este informe tan pronto como sea posible permite al equipo de desarrollo comenzar cuanto antes con las correcciones (si las hay) y seguir con el proyecto.
- El informe debe contener:
  - Un resumen de las discusiones en la revisión.
  - La decisión alcanzada en la reunión.
  - Una lista completa de todas las acciones necesarias (correcciones, cambios y añadidos).
  - Para cada punto de la lista se reflejan los responsables de llevar a cabo cada cambio.
  - Los nombres de los revisores encargados del seguimiento.

Revisiones Formales de Diseño: El seguimiento

- El encargado del seguimiento de las correcciones, en muchos casos el líder de la revisión, debe decidir si cada punto de la lista de modificaciones se ha cumplido para permitir que el proyecto continúe.
- El seguimiento también debe ser documentado, por si es necesaria cualquier aclaración en el futuro.
- Las modificaciones de pequeña envergadura no es necesario seguirlas en general, aunque alguien del equipo de desarrollo debe ser declarado responsable de su finalización.
- Igualmente, deberá asumir la responsabilidad si no se llevan a cabo.

Revisiones Formales de Diseño

Discusión

Suponed que, durante este curso, os pidiese realizar un proyecto conjunto entre todos los matriculados a la asignatura. ¿Cómo organizaríais el diseño y las revisiones formales?

Revisión por pares

- La mayor diferencia entre las revisiones formales y las revisiones por pares radica en los participantes.
- Mientras que en las revisiones formales hemos insistido en elegir al equipo entre los miembros con más experiencia y conocimientos, la revisión por pares la llevan a cabo gente del mismo nivel (es decir, pares).
- Además, la revisión por pares no tiene autoridad para evitar que el proyecto avance a la siguiente fase.
- Su objetivo es encontrar errores y diferencias con respecto a los estándares.

Revisión por pares

- Uno de los riesgos de la fase de diseño durante los últimos años es la disminución de las revisiones debida al uso de herramientas gráficas y herramientas de ayuda en general.
- Esto no significa que no deban usarse este tipo de herramientas, sino que deben usarse mientras se conservan las revisiones.
- Diversos estudios han comprobado que la revisión por pares sigue encontrando muchos errores.
- Hay 2 tipos de revisiones por pares, los tutoriales (*walkthrough*) y las inspecciones.

Revisión por pares

- La diferencia entre ambas es el nivel de formalismo.
- La inspección es la más formal de las dos.
- La inspección hace énfasis en las acciones correctivas.
- Es decir, mientras los tutoriales se limitan a comentar el documento revisado, las inspecciones también deben incluir cómo solucionar los problemas encontrados.
- Por tanto, se considera que las inspecciones son más importantes desde el punto de vista de la garantía de calidad.

Revisión por pares

- Para llegar a tener buenas inspecciones es necesario:
  - Crear listas de características que deben comprobarse según el tipo de documento, lenguaje de programación o herramienta, que se deben actualizar periódicamente.
  - Desarrollo de tablas con las frecuencias típicas de los defectos, para poder indicar a los inspectores donde se concentran los problemas.
  - Formar profesionales competentes en inspecciones, de tal manera que puedan actuar como líderes de inspección (moderadores) o como miembros del equipo. De esta manera estaremos formando inspectores para futuros proyectos.
  - Análisis de las inspecciones previas, para mejorar la metodología.
  - Planificar inspecciones dentro de las actividades del proyecto y reservar los recursos necesarios.

Revisión por pares: Participantes

- También en este caso el tamaño ideal del equipo es de entre 3 y 5 miembros.
- En general se debe incluir:
  - El líder de la revisión.
  - Uno de los autores.
  - Profesionales especializados.
- El autor es simplemente uno de los miembros del equipo de desarrollo.
- Puede aclarar las dudas que surjan sobre el diseño o el código.

Revisión por pares: Participantes

- Al líder de la revisión se le denomina moderador en las inspecciones y coordinador en los tutoriales.
- A pesar de los distintos nombres, sus funciones son básicamente las mismas en ambos casos.
- Sus obligaciones incluyen:
  - Tener buena formación en el desarrollo de proyectos en el área de la revisión y tener familiaridad con la tecnología que se esté usando.
  - Mantener una buena relación con el equipo de desarrollo, especialmente con la persona que participa en la revisión.
  - Ser externo al proyecto.
  - Tener experiencia en tareas de coordinación y dirección de reuniones profesionales.
  - En el caso de las inspecciones, tener cierta formación o experiencia como moderador.

Revisión por pares: Participantes

- Los requisitos que les pedimos a los profesionales en la revisión varían según el tipo de revisión.
- Para las inspecciones, se recomienda:
  1. Un diseñador. Se encargará de revisar el análisis y el diseño del sistema.
  2. Un implementador. En este caso es necesaria experiencia programando en el lenguaje y en el tipo de proyecto que se revisa. Se espera que detecte los defectos que pueden llevar a fallos.
  3. Un "tester". Un profesional con experiencia en testing, que se centra en buscar los errores que suelen aparecer en la fase de testing.

Revisión por pares: Participantes

- En el caso de los tutoriales, se recomienda que los profesionales se elijan buscando:
  1. Un experto en estándares. A este miembro del equipo se le asigna la tarea de localizar desvíos de los estándares y procedimientos fijados para el proyecto. Los errores en este campo suelen producir problemas a largo plazo, en primer lugar porque dificultan que se unan nuevos miembros al proyecto, y en segundo lugar porque complican el mantenimiento.
  2. Un experto en mantenimiento. Se encargará de buscar los defectos que puedan impedir corregir errores o puedan afectar al rendimiento en el futuro. Es deseable que tenga también experiencia en documentación, para poder revisar este aspecto.
  3. Un representante del cliente. Su función es contribuir a la validación del sistema desde el punto de vista del cliente. De no ser posible contar con un miembro de este equipo, se debe elegir a otro miembro del equipo que se encargue de simular esta función usando los requisitos.

Revisión por pares: Participantes

- Además de las tareas indicadas anteriormente, se debe asignar otras dos tareas entre los miembros del equipo.
- Alguien debe ejercer de presentador en las inspecciones.
- Este rol se lo asigna el moderador, y se encarga de explicar al resto del grupo los documentos que se están revisando.
- Es preferible en general que no sea el autor, ya que sus explicaciones pueden hacer más referencia a las ideas que tiene sobre el proyecto que a la realidad.
- Alguien debe ejercer de redactor.
- En general es el líder de la revisión, aunque no es obligatorio.
- Debe tomar nota de todos lo defectos que encuentren los miembros del equipo.
- No basta con hacer una lista de defectos, es necesario explicar bien todas sus características y riesgos.

Revisión por pares: Preparación

- El líder de la reunión debe:
  - Decidir qué documentos se deben revisar, discutiéndolo con el autor.
  - En general se deben revisar las funciones más complejas, críticas, o las que se sepa estadísticamente que pueden contener más errores.
  - Seleccionar los miembros del equipo.
  - Planificar las reuniones. En general es recomendable que cada sesión no supere las 2 horas, por lo que pueden ser necesarias varias reuniones si el proyecto es complejo.
  - Es también recomendable planificar las reuniones lo más cerca posible de la finalización de la fase, para no afectar mucho al desarrollo.
  - Distribuir los documentos al resto de miembros del equipo antes de las reuniones.
  - Decidir si es necesaria una reunión informativa previa, según el conocimiento que tengan los miembros del equipo sobre el proyecto.

Revisión por pares: Preparación

- Los miembros del equipo tienen más trabajo en las inspecciones que en los tutoriales.
- En las inspecciones deben asistir a la reunión previa (si la hay), en la que el autor explica el documento.
- Además, deben leer las secciones del documento que van a ser revisadas y escribir sus comentarios antes de la reunión.
- Esta preparación previa intenta maximizar la eficiencia de la reunión.
- Es interesante que usen una lista de características, si la hay, para analizar el documento.
- En los tutoriales, el equipo lee los documentos sin entrar en detalles, para tener una idea general.
- En lugar de la reunión previa, si alguien no es experto en el área debe dedicar más tiempo a leer el documento.
- Se pueden preparar los comentarios en las reuniones.

Revisión por pares: Reuniones

- En general la sesión empieza con una breve explicación del presentador sobre las secciones del documento a tratar.
- Después, los miembros del equipo presentan sus comentarios, o discuten los comentarios del resto.
- En los tutoriales, los comentarios están más basados en la presentación previa.
- Las discusiones se deben centrar en identificar errores, pero no dedicar tiempo a buscar soluciones.
- Durante la sesión, el redactor toma nota de cada defecto, apuntando su localización, descripción, tipo y carácter (incorrecto, ausente o innecesario).
- Debe también incluir una estimación de la severidad del defecto, que se usará más tarde con fines estadísticos.
- Como hemos dicho, no deberían durar más de 2 horas, ni haber más de 2 reuniones diarias.

Revisión por pares: Reuniones

- Hay dos documentos que es necesario generar en cada reunión:

  Informe de descubrimientos. Este documento, que hemos descrito en la transparencia anterior, debe completarse y distribuirse nada más acabar la sesión. Su objetivo es asegurar que todos los defectos identificados están completamente documentados para ser su futura corrección y seguimiento.

  Informe de resumen de la sesión de inspección. Este informe es redactado por el líder de la inspección después de la sesión (o sesiones) que tratan con un solo documento. En el resumen se incluyen los errores encontrados y los recursos necesarios (las horas que cada miembro ha dedicado) para ello. También puede presentar métricas de calidad y eficiencia. Este informe sirve para mejorar futuras revisiones.

Revisión por pares: Tareas tras una reunión

- Las tareas tras una reunión varían entre tutoriales e inspecciones.
- Mientras que el tutorial básicamente no tiene obligaciones entre reuniones, la inspección debe:
  - El autor debe trabajar junto con el resto del equipo de desarrollo para corregir los problemas indicados en el informe.
  - Transmitir los informes a los encargados de análisis, para que tomen medidas preventivas que eviten estos mismos defectos en el futuro.

Revisión por pares: Eficiencia

- Algunas de las métricas que se usan para calcular la eficiencia de estas revisiones son:
  - Número de horas trabajadas por defecto encontrado.
  - Número medio de defectos encontrados por página del documento analizado.
  - Porcentaje de defectos detectados en la revisión por pares con respecto al total de defectos detectados en total

Opiniones de expertos

- Este es el último tipo de revisión que veremos.
- La participación de expertos, que deben ser externos, permite añadir un punto de vista o experiencia en aspectos que no pueden cubrirse con el personal interno.
- Los expertos pueden:
  - Dar una opinión especializada sobre alguna parte del diseño, la documentación o el código.
  - Participar como un miembro más en una revisión interna, como las que hemos visto anteriormente.

Opiniones de expertos

Discusión

¿Qué ventajas ves a pedir opiniones de expertos? ¿Y desventajas?

Opiniones de expertos

- Las revisiones de expertos proporcionan los siguientes beneficios:
  - Experiencia en un área nueva para el equipo de desarrollo.
  - Apoyo cuando los miembros internos no pueden llevar la carga de trabajo debido a otros proyectos.
  - Este apoyo es interesante porque en otro caso el proyecto se paraliza hasta que dichos miembros vuelvan a estar disponibles, retrasando todo el proyecto (actúan como solución a un riesgo).
  - Mediación o "voto de calidad" cuando hay diferencias irreconciliables entre los expertos internos.
  - Ayuda en las revisiones en las compañías pequeñas.

¿Qué estamos viendo?

- Dentro de las actividades de calidad en el ciclo de vida del software tenemos:
  - La fase de desarrollo.
  - La fase de mantenimiento.
- Dentro de la fase de desarrollo, en la que estamos, tenemos:
  - Revisiones (formales y por pares).
  - Opiniones de expertos.
  - Testing.
- Hemos visto primero cómo es el proceso de desarrollo, el coste de la eliminación de errores, las revisiones y las opiniones de los expertos.
- Hablaremos brevemente sobre testing antes de pasar al mantenimiento.

Testing

- Como hemos visto, la fase de testing es muy importante dentro del desarrollo del proyecto y dentro del proceso de garantía de calidad.
- Durante esta fase debemos probar la implementación, y comprobar que los resultados se ajustan a los requisitos.
- El gran problema del testing es que no es completo, es decir, que nuestras prueban no encuentren un error no significa que no exista.

Testing

Ejemplo

Supongamos un programa de ordenación de listas (es decir, muy sencillo) que solo falla si en la lista inicial hay dos elementos iguales seguidos y la longitud de la lista es mayor de 10.
Es muy posible que hagamos varias pruebas y no encontremos el error.

Testing

- Además de estos errores, que pueden parecer "artificiales", hay cosas que simplemente no se pueden comprobar.
- Por ejemplo, no podemos asegurar que un programa termine por muchos casos de test que utilicemos.
- En la fase de testing vuelven a ser muy importantes los requisitos.
- Es posible que haya "errores" en funciones que se deban a que el usuario no dijo qué ocurría cuando la entrada tomaba ciertos valores.
- Como ya hemos visto, esto supone un gran coste.

Testing

Ejemplo (Conjetura de Collatz)

Dado el siguiente programa:

```
public static void collatz(int n) {
  if (n == 1) return;
  else if (n % 2 == 0) collatz(n / 2);
       else collatz(3*n + 1);
}
```

Nadie ha sido capaz de demostrar que no termine, y ha funcionado para todos los números que se ha probado (hasta varios billones).
¿Fallará justo para el número que introduzca el cliente?

Testing

- Hay varios tipos de testing:

  Caja negra. En este tipo de testing solo conocemos los requisitos, y hacemos pruebas para ver si obtenemos los resultados esperados.

  Caja de cristal/blanca. En este tipo de testing conocemos la implementación, y la usamos para ver si falla.

- Dentro del testing de caja blanca hay varias estrategias.
- Veremos el tema del testing con detalle en el último tema.

Testing: Otros métodos

- Si quisiéramos asegurar que algo nunca falla es necesario recurrir a algo más potente que el testing: la demostración de teoremas.
- El problema en este caso es la gran complejidad de la tarea y la falta de formación, que hace que pocas personas dentro de una empresa tengan la capacidad de llevarla a cabo.
- En la actualidad, algunas aplicaciones llevan una demostración de propiedades críticas, que el procesador a cargo de ejecutarla "repasa" antes de la ejecución.
- Por ejemplo, algunas aplicaciones móviles prueban que no consumen más que una cierta cantidad de memoria o procesador.
- Esta técnica se conoce como proof-carrying code.

Testing: Otros métodos

- Una importante técnica para comprobar la corrección de un modelo a partir de una cierta configuración de la entrada es la comprobación de modelos (model checking).
- El model checking es, a diferencia de la demostración de teoremas, automático.
- Podemos escribir fórmulas en distintas lógicas, según el model checker que estemos usando.
- En general suelen ser propiedades temporales del estilo "alguna vez ocurre algo bueno", "nunca ocurre nada malo".

Testing: Otros métodos

- Por ejemplo, si estamos definiendo una sección crítica querríamos decir "si un proceso solicita entrar en la sección crítica, acaba entrando" o "ningún proceso monopoliza la sección crítica".
- Estas propiedades se comprueban para unos datos de entrada concretos, no para todos los casos.
- Como en el caso del testing, no podemos asegurar que *siempre* se cumpla.
- Estas propiedades son muy interesantes cuando hay concurrencia, como la que introducen los hilos.
- El model checking se puede combinar con la generación automática de casos de test.
- Veremos con detalle la técnica de model checking en el apartado de Fiabilidad.

Contenidos

Actividades de calidad durante el mantenimiento

- En el ciclo de vida del software la fase de desarrollo puede llevar meses, aunque puede llegar a años en productos grandes como sistemas operativos.
- El periodo durante el cual se usa el producto es mucho mayor, puesto que se extiende como mínimo durante años, en ocasiones durante décadas.
- Por este motivo, es necesario planificar cuidadosamente la fase de mantenimiento.
- En esta parte del tema veremos cómo garantizar la calidad durante esta fase.

Actividades de calidad durante el mantenimiento

- Podemos distinguir tres elementos durante la fase de mantenimiento:

  Mantenimiento correctivo. Corrige los errores que el usuario encuentra en el software o en los servicios.

  Mantenimiento adaptativo. Modifica el software si el cliente añade nuevos requisitos, el entorno cambia (e.g. aparece un nuevo sistema operativo), etc.

  Mantenimiento para mejorar la funcionalidad. En realidad es la combinación de otros dos tipos de mantenimiento: el mantenimiento de perfeccionamiento, que añada nuevas funciones para mejorar el rendimiento, y el mantenimiento preventivo, que mejoran la confiabilidad y la infraestructura del sistema para facilitar el mantenimiento futuro.

Actividades de calidad durante el mantenimiento

- Cada elemento tiene un objetivo distinto.
- El mantenimiento correctivo asegura que el cliente puede utilizar el software tal y como se había especificado.
- El mantenimiento adaptativo trata de ampliar la cantidad de usuarios que pueden usar el producto.
- El mantenimiento para mejorar la funcionalidad pretende alargar el periodo de vida del software.

Actividades de calidad durante el mantenimiento

- Los errores que encuentre el cliente se pueden deber a:
  - En error en el código (un fallo software).
  - Un error en el manual del usuario, los formularios de ayuda o cualquier otro tipo de documentación destinada para el usuario. En ciertos casos puede bastar con explicar las instrucciones correctas al usuario, sin necesidad de modificar el software ni la documentación.
  - Documentación vaga, incompleta o imprecisa.
  - La falta de conocimientos del usuario. En este caso no podemos solucionarlo.

Actividades de calidad durante el mantenimiento

- Obviamente, deberemos trabajar sobre los 3 primeros puntos, que se refieren a mantenimiento correctivo.
- En general es necesaria la participación del usuario en estos casos.
- También puede serlo para el adaptativo.
- Sin embargo, para mejorar la funcionalidad no es necesaria la participación del usuario.
- En general, se calcula que el 60 % de los recursos invertidos en diseño y programación en un proyecto se dedican al mantenimiento.

Actividades de calidad durante el mantenimiento

- La garantía de calidad durante el mantenimiento tiene tres objetivos principales:
  1. Asegurar, con un cierto nivel de confianza, que las actividades de mantenimiento se ajustan a los requisitos técnicos.
  2. Asegurar, con un cierto nivel de confianza, que las actividades de mantenimiento se ajustan a los plazos y el presupuesto.
  3. Iniciar y controlar las actividades para mejorar e incrementar la eficiencia del mantenimiento del software y las actividades de calidad. Esto supone alcanzar los objetivos mientras se reducen costes.

Actividades de calidad durante el mantenimiento: Factores

- La fase de mantenimiento debe conservar la calidad que el software original tenía, o mejorarla cuando se encuentre algún error.
- También debemos verlo al revés: si no se tuvo cuidado en garantizar la calidad (el mantenimiento) durante el desarrollo, entonces esta fase será cara y poco eficiente.
- Por ello, vamos a revisar los factores del software que son necesarios para poder tener un buen mantenimiento.
- Como recordaréis, los factores eran las áreas que se debían concretar en la fase de especificación de requisitos para asegurar la calidad.

Actividades de calidad durante el mantenimiento: Factores

Corrección. No solo es importante que las salidas especificadas por el usuario den los valores adecuados, también debemos hablar de la corrección de la documentación y de cómo se ajusta la implementación a los estándares.

Confiabilidad. Cuanto menor sea más veces contactará el cliente con nosotros.

Mantenibilidad. Obviamente, este es el factor clave que nos permite que la fase de mantenimiento sea eficiente.

Flexibilidad. Los esfuerzos en este factor supondrán mejoras en el mantenimiento adaptativo.

Actividades de calidad durante el mantenimiento: Factores

Testabilidad. Si permitimos que el sistema pueda recibir entradas en distintos formatos y a diferentes niveles del sistema, podemos probar directamente los errores que haya sufrido el usuario (que los puede enviar simplemente haciendo click en una ventana que le pregunte si desea enviar la causa del error). De esta manera se simplifica el proceso de detección del error.

Portabilidad. Simplifica el mantenimiento adaptativo.

Interoperabilidad. Simplifica el mantenimiento adaptativo.

Actividades de calidad durante el mantenimiento: Política de mantenimiento

- Hay dos aspectos importantes a tener en cuenta a la hora del mantenimiento:
  - El control de versiones.
  - Las políticas de cambio.
- El control de versiones básicamente se preocupa por cuantas versiones del producto se mantienen al mismo tiempo.
- Las políticas de cambio estudian cada solicitud de cambio y deciden si aprobarlo o no.

Actividades de calidad durante el mantenimiento: Control de versiones

- El control de versiones se complica cuanto mayor sea la cantidad de usuarios.
- Las versiones pueden desarrollarse de manera secuencial o de árbol.
- Las versiones secuenciales solo permiten usar una versión a todos los usuarios.
- Como hay usuarios de distintos perfiles, mucha funcionalidad puede ser redundante.
- Cada nueva versión simplemente sustituye a la anterior.

Actividades de calidad durante el mantenimiento: Control de versiones

- Las políticas de versiones en árbol desarrollan versiones específicas del software para un grupo de usuarios.
- Las nuevas versiones tiene en cuenta solo las solicitudes de cambio de cada grupo concreto de usuarios.
- Cada "rama" de la aplicación difiere en complejidad y funcionalidad.
- El mantenimiento en este caso es más caro y complejo, pues hay varias versiones a la vez.

Actividades de calidad durante el mantenimiento: Control de versiones

Ejemplo (Inventarios)

Supongamos que tenemos un programa para administrar inventarios. Tras unos años con versiones en árbol, lo hemos especializado para 7 clientes: farmacias, electrónica, hospitales, librerías, supermercados, talleres y plantas químicas.
Cada rama tiene tiene algunas ramas secundarias en las que varían algunos módulos y el nivel de implementación de ciertas aplicaciones.
Por ejemplo, la versión para librerías tiene 3 subramas: librerías con franquicias, librerías con una sola tienda y librerías universitarias.

Actividades de calidad durante el mantenimiento: Control de versiones

Ejemplo (Inventarios)

En total, el equipo de mantenimiento da servicio a 30 versiones diferentes del programa al mismo tiempo, mejorando además cada una de ellas cuando lo solicitan los clientes.
Esto les lleva a ciertas dificultades que no se dan en general en otros casos:

- *Problemas del usuario al identificar la versión con la que están teniendo errores.*
- *Errores al usar el código de otro módulo para arreglar un mismo problema. Es posible que las funciones relacionadas no funcionen igual en todas las versiones.*
- *Problemas al insistir a los usuarios que usen otra versión del producto. Si algún usuario usa una versión anticuada de una rama los problemas se multiplican.*

Actividades de calidad durante el mantenimiento: Control de versiones

Ejemplo (Inventarios)

Por estos motivos, el jefe del servicio de mantenimiento cree que se debería haber seguido una política secuencial, de tal manera que ofrecería opciones configurables a sus usuarios para que lo personalicen para su negocio.
De esta manera, supondría mucho menos trabajo mantener el sistema.
¿Qué ventajas concretas ves?
Sin embargo, ¿cuáles serían las desventajas?

Actividades de calidad durante el mantenimiento: Control de versiones

Discusión

¿Conoces algún programa que tenga distintas versiones en árbol?

Actividades de calidad durante el mantenimiento: Políticas de cambio

- Hay que ser cauto con las políticas de cambio, pues una política demasiado permisiva supone un incremento, en muchas ocasiones injustificado, de la carga de trabajo.
- Es mucho más conveniente tener un política equilibrada, que aprueba los cambios solo después de un examen exhaustivo.
- Esto supone dedicar más recursos a esta tarea.
- Sin embargo, permite ahorrar en los cambios, por lo que en general resulta mejor a largo plazo.

Ejemplo

Google Mail estuvo tanto tiempo en versión beta que cuando salió la versión estable ofrecían una opción para conservar la palabra "Beta" en el logo.

Actividades de calidad durante el mantenimiento: Componentes previos

- De la misma manera que debemos comprobar ciertos detalles de calidad antes de comenzar el desarrollo, son necesarias ciertas comprobaciones para el correcto funcionamiento del mantenimiento:
  - La revisión del contrato de mantenimiento.
  - El plan de mantenimiento.
- De esta manera podemos asegurar que el mantenimiento conserva la calidad y se ajusta a los plazos y presupuestos.

Actividades de calidad durante el mantenimiento: Contrato

- Se debe tener una perspectiva amplia cuando se contrata el mantenimiento.
- En concreto, se deben especificar las categorías para las que el mantenimiento va a ser necesario.
- Estas dependen del cliente: software específico, software general o clientes internos.
- Aunque es aconsejable seguir las ideas que ya hemos visto para revisar el contrato de desarrollo, hay algunos aspectos más específicos.

Actividades de calidad durante el mantenimiento: Contrato

1. Aclaración de requisitos
   - El tipo del mantenimiento correctivo que se desea: consultas on-line y consultas personales, horas de servicio, tiempo de respuesta, etc.
   - Cantidad de usuarios de la aplicación.
   - Localización de los usuarios.
   - Necesidad de mantenimiento adaptativo y de mejora.
2. Revisión de los recursos necesarios Primero se deben estudiar los recursos que necesitará el mantenimiento, y después la empresa debe comprobar que puede proporcionarlos (es posible que sea necesario dedicar empleados simplemente a estas tareas, que por tanto ya no podrán desarrollar).

Actividades de calidad durante el mantenimiento: Contrato

3. Alternativas al mantenimiento
   - Posibilidad de subcontratar el servicio.

Discusión

¿Qué problemas plantea esta opción?

   - Posibilidad de que el usuario resuelva los problemas con asistencia.
4. Revisión del mantenimiento de aplicaciones desarrolladas por subcontratas Se debe asegurar que serán ellos quienes se ocuparan del mantenimiento. En caso contrario, se debe tener en cuenta en el correspondiente contrato.
5. Revisión de plazos y presupuestos

Actividades de calidad durante el mantenimiento: Contrato para clientes internos

- Es frecuente que no se realice contrato para los clientes internos.
- Lo más común en estos casos es que simplemente se proporcione mantenimiento continuado.
- Esto puede producir problemas por ambas partes.
- Por un lado el cliente parece que esté solicitando un "favor" cuando se necesita mantenimiento.
- Por otro lado, el desarrollador se siente "molesto" cuando debe abandonar otro proyecto.
- Para evitar esto se debe escribir un contrato de servicio interno.
- En él se deben indicar claramente los servicios que se proporcionarán en el futuro.
- De esta manera se evitan posteriores problemas.

Actividades de calidad durante el mantenimiento: Plan de mantenimiento

- El plan de mantenimiento proporciona un marco para organizar el mantenimiento.
- En él deben participar todos los clientes.
- Está basado en las mismas ideas que el plan de desarrollo que ya hemos visto, pero centradas en el mantenimiento.
- El plan incluye:
  1. El presupuesto del mantenimiento.

Actividades de calidad durante el mantenimiento: Plan de mantenimiento

2. Una lista de los servicios contratados. Se refiere a la aclaración de requisitos comentada anteriormente.
3. La descripción de la organización del equipo de mantenimiento. Se refiere principalmente a las necesidades de personal, que se fijan siguiendo estos criterios:
   - La cantidad de personal necesaria. Si además se va a distribuir el personal en distintas localizaciones, la cantidad necesaria en cada lugar.
   - Los conocimientos necesarios de cada equipo.
   - Organización de los equipos, indicando al líder.
   - Definición de las tareas de cada equipo.
   - Los requisitos de formación.

Actividades de calidad durante el mantenimiento: Plan de mantenimiento

4. La infraestructura necesaria, que incluye:
   - La localización desde la cual se proporcionará el mantenimiento, y que requiere el hardware necesario para ejecutar la aplicación y equipamiento para comunicarse con el cliente.
   - Un centro de documentación desde el que sean accesibles los siguientes documentos (no necesariamente en formato físico):
     - La documentación del software.
     - El contrato de servicios.
     - El software que cada cliente tiene instalado y su configuración.
     - El historial de mantenimiento para cada usuario.

Actividades de calidad durante el mantenimiento: Plan de mantenimiento

5. Una lista de los riegos de mantenimiento encontrados. Los riesgos de mantenimiento se refieren a las situaciones en las que un fallo imposibilita que se proporcione mantenimiento. Estos riesgos incluyen:
   - Reducciones de plantillas, tanto en la organización de los servicios, en una localización específica o para una aplicación específica (e.g. es un riesgo tener un solo experto en un área, porque nos deja sin alternativas en caso de abandonar la empresa).
   - Insuficiente formación o familiaridad con ciertas partes del producto, que retrasan el tiempo de respuesta y las correcciones.
   - Cantidad insuficiente de personal para mantenimiento adaptativo o de mejora si el cliente necesita un cambio de envergadura.

Actividades de calidad durante el mantenimiento: Plan de mantenimiento

6. Una lista de los procedimientos y controles necesarios para el mantenimiento.
   Los procedimientos se refieren a los procesos que llevan a cabo los equipos de mantenimiento correctivo y los que deben tratar directamente con el cliente. Estos procedimientos tratan en general con:
   - Cómo tratar un informe de error por parte del usuario.
   - Cómo tratar las solicitudes del usuario.
   - Informes periódicos y seguimiento de los servicios de ayuda al cliente.
   - Informes periódicos y seguimiento de las actividades de mantenimiento correctivo.
   - Formación y acreditación del personal.

Actividades de calidad durante el mantenimiento: Herramientas

- Una gran variedad de herramientas se usan durante la fase de mantenimiento.
- Los distintos tipos de mantenimiento que hemos visto requieren distintos tipos de herramientas.
- Es preciso indicar que "herramientas" no solo se refiere a programas que puedan facilitar estas tareas, también a métodos para asegurar la calidad en esta fase.

Actividades de calidad durante el mantenimiento: Herramientas para el mantenimiento correctivo

- Las herramientas en este caso se dividen en dos grupos:
  - Herramientas para interactuar con el usuario.
  - Herramientas para realizar las correcciones.
- La interacción con el usuario es necesaria cuando se encuentran errores en el código o en la documentación.
- También es necesaria cuando la documentación es vaga o incompleta.
- Además de en los casos de fallo, el usuario puede querer contactar con el servicio técnico cuando sus conocimientos no le permiten usar una función.

Actividades de calidad durante el mantenimiento: Herramientas para el mantenimiento correctivo

- Las correcciones del primer tipo son más frecuentes al principio.
- Se siguen dando después, pero con menor frecuencia.
- Se usan distintas herramientas según los errores sean de software o de documentación.
- Aunque se usen herramientas diferentes, el mismo equipo suele estar a cargo de todo el mantenimiento.

Actividades de calidad durante el mantenimiento: Herramientas para el mantenimiento correctivo

- Un tipo especial de herramientas útil para arreglar errores son las herramientas del "mini ciclo de vida de la garantía de calidad".
- La principal de estas herramientas es el mini-testing.
- El mini-testing se usa para tareas cortas de corrección, y se caracteriza por por un número pequeño de cambios en muy poco tiempo.
- Dado que hay poco tiempo para realizar los cambios, también hay poco para el testing, por lo que se intenta realizar unos test mínimos antes que ninguno.

Actividades de calidad durante el mantenimiento: Herramientas para el mantenimiento correctivo

- La garantía de calidad del mini-testing requiere:
  - El testing lo lleva a cabo un especialista, no el programador.
  - Se debe preparar un documento del procedimiento de testing, de unos 2-3 páginas.
  - En este documento se incluye una descripción de los efectos esperados de la reparación, el alcance de las correcciones y una lista de los casos de test que se usarán.
  - También se debe preparar un documento de re-testing, parecido al anterior, para tratar con los errores que se encuentren con los tests anteriores.
  - Se debe redactar un informe documentando los errores encontrados en el testing y el re-testing.
  - El jefe del equipo de testing debe revisar estos documentos y dar su conformidad a todos los puntos.
  - Si los cambios son triviales se puede incluso prescindir del mini-testing.

Actividades de calidad durante el mantenimiento: Herramientas para el mantenimiento correctivo

- Subcontratar el mantenimiento se está volviendo una práctica común.
- En la mayoría de los casos porque es más sencillo y económico que hacerlo uno mismo.
- La mejor manera de seguir asegurando la calidad es trabajando en la relación empresa-subcontrado.
- Para ello, debemos dejar claro en el contrato que son necesarios:
  - Procedimientos para tratar con un cierto rango de solicitudes de mantenimiento.
  - Total documentación de los procedimientos de servicio.
  - Disponibilidad de registros que documenten la capacidad del equipo de mantenimiento.
  - Autorización para poder realizar revisiones y encuestas de satisfacción.
  - Posibilidad de imponer multas o el fin del contrato si no se cumplen las condiciones.

Actividades de calidad durante el mantenimiento: Herramientas para el mantenimiento adaptativo y de mejora

- Esta parte del mantenimiento es muy similar a las tareas de desarrollo de software.
- Por ello, se suelen utilizar las medidas ya contadas para este caso (revisiones y testing).
- Por las mismas razones, estos son también los métodos utilizados para llevar a cabo el mantenimiento adaptativo.

Actividades de calidad durante el mantenimiento: Infraestructuras

- Las herramientas sobre infraestructura son esenciales en el mantenimiento del software
- Muchas de ellas tienen un carácter general y se usan durante todo el ciclo de vida.
- Sin embargo, también son necesarias ciertas infraestructuras específicas para el mantenimiento.
- Como ya hemos visto, el uso de estas herramientas debe comenzar en la fase de desarrollo, para permitir un mantenimiento eficiente.
- Es decir, dentro de estas herramientas consideramos tanto las que ayudan a desarrollar software de calidad como aquellas que permiten mantener el producto final.

Actividades de calidad durante el mantenimiento: Infraestructuras

- Las herramientas para infraestructuras son especialmente importantes para mantenimiento correctivo.
- Nos concentraremos en las siguientes herramientas:
  1. Procedimientos de mantenimiento.
  2. Dispositivos para calidad.
  3. Formación y certificación del personal.
  4. Acciones preventivas y correctivas.
  5. Administración de configuraciones.
  6. Control de la documentación y de los registros de calidad.

Actividades de calidad durante el mantenimiento: Infraestructuras

1. La mayoría de los procedimientos de mantenimiento se aplican, como hemos dicho, al mantenimiento correctivo y a actividades de ayuda al cliente, como:
   - Ayuda remota en caso de fallo.
   - Ayuda presencial en caso de solicitarlo el cliente.
   - Servicio de atención al cliente.
   - Certificación del mantenimiento y del equipo de atención.
   - Garantía de calidad del mantenimiento y del equipo de atención.
   - Encuestas de satisfacción.

Actividades de calidad durante el mantenimiento: Infraestructuras

2. El departamento de mantenimiento debe desarrollar dispositivos especializados para corregir el software y atender al cliente.

- Dichos dispositivos incluyen plantillas, listas de características y similares.
- En concreto, se debe incluir:
  - Listas en las que se indique la localización de los fallos.
  - Listas de características para preparar un procedimiento de mini-testing.
  - Plantillas para informar de cómo se han solucionado los fallos software.
  - Estas plantillas deben incluir información sobre la manera en que se detectaron los fallos.

Actividades de calidad durante el mantenimiento: Infraestructuras

3. La formación de los equipos de mantenimiento adaptativo y para mejoras es similar a la formación de los equipos de desarrollo.

- Sin embargo, los encargados del mantenimiento correctivo sí que necesitan una formación específica.
- La formación de este tipo de personal está motivada por los términos del contrato de mantenimiento, que hace que este servicio sea necesario de manera continuada.
- El plan de formación debe incluir también la necesidad de formar personal extra para apoyo cuando haya picos de trabajo o para reemplazar a aquellos que abandonan el equipo.
- La certificación en corrección de software y para personal de servicio técnico deben tener en cuenta las características propias de estos trabajos.
- Se debe prestar especial atención al hecho de que estas personas trabajan normalmente bajo grandes presiones de plazo, solos y en una localización distinta a la del resto del equipo.

Actividades de calidad durante el mantenimiento: Infraestructuras

4. Acciones preventivas y correctivas.

- Durante el proceso de desarrollo se genera información muy valiosa para la fase de mantenimiento.
- Esta información incluye los registros de fallos software y cómo se solucionaron.
- Usando los registros de pasados proyectos podemos prever errores y solucionarlos, así como encontrar soluciones para problemas en nuestro propio proyecto.
- Los registros del proyecto actual también son importantes, pues indican cómo se afrontaron ciertos problemas que pueden persistir.
- Para que este proceso sea efectivo, es necesario disponer de procedimientos para almacenar la información de tal manera que se puede recuperar de manera sencilla.
- También es necesario que sea posible añadir información para prevenir estos problemas en el futuro, o sobre nuevos problemas y soluciones.

Actividades de calidad durante el mantenimiento: Infraestructuras

- Estas actividades deben ser dirigidas y controladas por un comité interno.
- Este comité, llamado Comité de Acciones Preventivas, suele estar presente en todas las grandes empresas.
- Los asuntos que suelen ser enviados al comité incluyen:
  - Cambios en el contenido y la frecuencia de las solicitudes del usuario al servicio técnico.
  - Mayor tiempo empleado en contestar las solicitudes del cliente.
  - Mayor tiempo empleado en arreglar los fallos.
  - Mayor porcentaje de fallos tras la corrección.

Actividades de calidad durante el mantenimiento: Infraestructuras

5. Administración de configuraciones.

- El equipo de mantenimiento depende en gran medida de la administración de configuraciones.
- Esta dependencia se debe a que el software, como hemos visto, es usado durante años, y en este periodo las versiones van cambiando, aunque muchos usuarios no actualizan el software o utilizan versiones en árbol.
- Dos actividades comunes que dependen de las configuraciones son los errores de corrección y los reemplazamientos en grupo de la versión actual a una nueva debidos al equipo de mantenimiento.
- En el caso de los errores de corrección, se necesita información del tipo:
  - La versión del software instalado cuando se produjo el error.
  - Una copia del código y su documentación.

Actividades de calidad durante el mantenimiento: Infraestructuras

- En los reemplazamientos en grupo el termino "grupo" se refiere a todos aquellos usuarios que tienen la misma versión del software instalada.
- Por tanto, reemplazamiento en grupo significa que todos estos usuarios recibirán la nueva versión al mismo tiempo.
- La administración de configuraciones se debe encargar, en este caso, de:
  - Tomar la decisión de cuándo hacer el reemplazo, teniendo en cuenta la cantidad de usuarios y de sus contratos.
  - Planificar los recursos necesarios y determinar los plazos.
- Una buena administración impide que se generen problemas de versiones y por tanto minimiza la necesidad de apoyo técnico.
- También es útil cuando la nueva versión soluciona un error común, pues evita que muchos usuarios soliciten ayuda para el mismo problema.

Actividades de calidad durante el mantenimiento: Infraestructuras

6. Control de la documentación y de los registros de calidad.

- Los requisitos específicos para este punto están íntimamente relacionados con el mantenimiento de corrección y de servicio técnico.
- La documentación y los registros se preparan de acuerdo a:
  - Proporcionar datos para correcciones preventivas, como hemos explicado anteriormente.
  - Facilitar la respuesta a futuros informes de fallo por parte del cliente.
  - Proporcionar evidencia de los cambios en caso de quejas o reclamaciones por parte del cliente.
- La documentación preparada debe tener en cuenta todos los aspectos listados.

Actividades de calidad durante el mantenimiento: Costes

- Vamos a centrarnos en el coste del mantenimiento correctivo.
- Los costes de calidad de este mantenimiento se clasifican en 6 clases:

  Costes de prevención. Incluyen el coste de formación del personal y de las actividades preventivas.

  Costes de evaluación. Incluyen el coste de detectar el error

  Costes de administración y control. Incluyen el coste de preparar el plan de matenimiento, de formación del personal y del seguimiento.

  Costes de fallo interno. Se refieren a las correcciones de errores encontrados por el equipo de mantenimiento (sin intervención del cliente).

  Costes de fallo externo. Se refieren a las correcciones de errores encontrados por el cliente.

  Costes de fallo de administración. Incluyen los costes por errores debidos por errores en la administración, como pueden ser falta de personal o personal con una formación insuficiente.

Actividades de calidad durante el mantenimiento: Costes

- En el caso del fallo externo, debemos hacer ciertas distinciones.
- Los costes son distintos si el error se produce mientras la garantía está en vigor o después.
  1. En el caso de correcciones software.
     - Todos los costes de corrección de software iniciados por el cliente son costes externos y el desarrollador debe asumirlos.
     - Una vez la garantía ha finalizado, las correcciones se consideran parte del servicio normal y no como costes de calidad.
     - Por tanto, durante este periodo solo las re-correcciones (correcciones de algo que se suponía que habíamos arreglado) suponen coste.

Actividades de calidad durante el mantenimiento: Costes

2. En el caso del servicio técnico
   - Mientras la garantía está vigente estos gastos se consideran de formación, y no costes de fallo externo.
   - Una vez acabada la garantía, las consultas son parte del servicio normal.
   - En ambos casos, solo consideramos fallo externo que el usuario tenga que repetir una consulta porque no se solucionó correctamente la primera vez.

- Como hemos visto en otros casos, la información que obtenemos del mantenimiento debe ayudarnos a tomar decisiones en futuros proyectos sobre:
  - Los puntos débiles de la calidad, donde se debe invertir para evitar pérdidas.
  - Desarrollo de una nueva versión del producto (en el caso de ser software por encargo) o sustitución del software.

Herramientas CASE en el mantenimiento

- También durante el mantenimiento podemos hacer uso de herramientas CASE específicas.
- En el caso del mantenimiento correctivo:
  - Usando la documentación generada automáticamente podemos hacernos una idea del funcionamiento del sistema y encontrar el fallo (en caso de no haber documentado correctamente antes).
  - Consultas en bases de datos nos permiten identificar mejor los errores y anticipar los efectos de las correcciones.
  - Uso de generadores automáticos de código, que es de esperar que no contengan ningún error, así como generadores de documentación (para las nuevas funciones). Usando el código correcto podemos hacernos una idea del error en nuestro código.

Herramientas CASE en el mantenimiento

- En el caso del mantenimiento adaptativo:
  - Una documentación completa y actualizada por herramientas CASE permite examinar posibles adaptaciones del software para nuevos usuarios o aplicaciones.
- En el caso del mantenimiento de mejora:
  - El uso de repositorios permite asegurar la consistencia de las nuevas aplicaciones.
  - Como en el caso de las correcciones, las consultas en bases de datos nos permiten planear cambios y añadidos conociendo previamente sus efectos.
  - Los cambios llevados a cabo con herramientas CASE permiten generar código automáticamente. Dado que este código no suele contener errores, podemos usarlo como prototipo.

Contenidos

Control de la documentación

- Como hemos visto hasta ahora, el desarrollo y el mantenimiento producen grandes cantidades de documentación.
- Algunos de los documentos generados deben ser distribuidos inmediatamente para permitir que el proyecto se desarrolle con normalidad.
- Por tanto, son necesarios procedimientos especiales de control de la documentación, llamados procedimientos de documentación para indicar qué documentos son necesarios en cada momento y asegurar así su preparación y disponibilidad.
- Los documentos que guardan estas características se llaman documentos controlados.

Control de la documentación

- Los registros de calidad son uno de dichos documentos controlados.
- Su objetivo es proporcionar pruebas que permitan comprobar que el desarrollo y el mantenimiento se llevan a cabo siguiendo los requisitos.
- También comprueban que el sistema de calidad se está siguiendo de manera efectiva.
- Varios estándares de ISO e IEEE se encargan de formalizar este tipo de documentación.

Documentos controlados y registros de calidad

Definición (Documento controlado)

Un documento controlado *es un documento que tiene o puede tener vital importancia bien para el desarrollo o el mantenimiento de un sistema software, bien para la administración de las relaciones actuales o futuras para el cliente. Por tanto su preparación, almacenamiento, recuperación y eliminación deben estar controladas por procedimientos de documentación.*
Los objetivos de los documentos controlados son:

- *Asegurar la calidad del documento.*
- *Asegurar la complección técnica y la adherencia a los procedimientos e instrucciones.*
- *Asegurar la disponibilidad futura de documentos que pueden ser necesarios para el mantenimiento, el desarrollo futuro o para las posibles quejas del usuario.*
- *Ayudar en la investigación de las causas que produzcan un fallo software y asignar la responsabilidad como parte de una acción correctiva.*

Documentos controlados y registros de calidad

Definición (Registro de calidad)

Un registro de calidad *es un tipo especial de documento controlado. En concreto, es un documento controlado dirigido al cliente que puede ser necesario para demostrar el total seguimiento de los requisitos y de la garantía de calidad a lo largo de todo el proceso de desarrollo y mantenimiento.*

Ejemplos de documentos controlados

- Documentos del ante-proyecto:
  - Informe de revisión del contrato.
  - Contrato de desarrollo.
  - Contrato de mantenimiento.
  - Contratos con las subcontratas.
  - Plan de desarrollo software.

Discusión

¿Quién puede necesitar estos documentos? ¿Por qué?

Ejemplos de documentos controlados

- Documentos del ciclo de vida del software:
  - Especificación de requisitos.
  - Diseño.
  - Descripción de las bases de datos.
  - Manual de usuario
  - Manual de mantenimiento.
  - Planes de instalación.
  - Cambios de requisitos.

Discusión

¿Quién puede necesitar estos documentos? ¿Por qué?

Procedimientos de control de documentación

- Para asegurar la calidad en la documentación tenemos los procedimientos de control de documentación.
- Estos procedimientos incluyen:
  - Definición de una lista con los tipos de documentos (y sus actualizaciones) que deben ser controlados.
  - Requisitos para la preparación de documentos.
  - Requisitos para la aprobación de documentos.
  - Requisitos para el almacenamiento y recuperación, incluyendo distintas versiones, revisiones y eliminaciones.

Procedimientos de control de documentación

- Las necesidades de documentación pueden variar entre distintas compañías.
- Algunas tareas de control de documentación, como el almacenamiento y la recuperación, son necesarias en cualquier caso.
- Hay muchos tipos de software para tratar con estas tareas, es importante seguir dedicando esfuerzos a ello, ya que la documentación es una parte integral de la garantía de calidad.
- También es importante recordar que se debe controlar la documentación de los servicios subcontratados.

Procedimientos de control de documentación: La lista de documentos controlados

- La lista de documentos controlados es la clave para el control de la documentación.
- Para construirla se debe establecer una autoridad que la implemente.
- Esa autoridad puede ser una sola persona o un comité.
- Esta autoridad es responsable de:
  - Decidir qué tipo de documento debe ser controlado y cuáles deben ser además registros de calidad.
  - Decidir si el nivel de control es el adecuado.
  - Hacer el seguimiento para comprobar que solo los documentos de los tipos definidos arriba son declarados como controlados.
  - Hacer el seguimiento para iniciar cambios, añadidos o eliminaciones en los tipos de documentos controlados.
- Es importante notar que parte de los documentos controlados son documentos de uso interno.

Procedimientos de control de documentación: Preparación

- Los requisitos de documentación en la creación de un nuevo documento o en la revisión de uno existente se centran en la completitud, legibilidad y disponibilidad.
- Estos requisitos se logran mediante:
  - Estructura.
  - Método de identificación.
  - Información de referencias y estándares.
- La estructura del documento puede ser libre o definida por una plantilla.
- El método de identificación pretende asignar un identificador único a cada documento, versión y revisión.
- Este método suele usar notación del (a) nombre o número del producto, (b) el tipo de documento y (c) la versión y el número de revisión.

Procedimientos de control de documentación: Preparación

- La información de referencias y estándares permite futuros accesos a los documentos a base de dar información sobre su contenido.
- Dependiendo del tipo de contenido, deberemos dar uno o más de los siguientes puntos:
  - El autor del documento.
  - La fecha de terminación.
  - Las personas que aprobaron el documento, incluyendo su puesto.
  - Fecha de aprobación.
  - Firmas del autor y de las personas que lo aprobaron.
  - La descripción de los cambios introducidos.
  - La lista de versiones y revisiones anteriores.
  - La lista de circulación (personas a quien debe ser enviado).
  - Restricciones de confidencialidad.
- Los procedimientos y las instrucciones de trabajo relevantes se pueden distribuir tanto de manera física como electrónica.

Aprobación de documentos controlados

- En ciertos casos los documentos controlados requieren aprobación.
- La aprobación no es solo necesaria para garantizar la calidad, también para asegurar que el documento se ajusta a los estándares y plantillas asignadas.
- En los casos en los que es necesaria la aprobación, los procedimientos indican el puesto de la(s) persona(s) que debe darla para cada tipo de documento.
- La aprobación puede depender de una persona o un comité.
- Además de tener el puesto indicado, se espera que las personas a cargo tengan experiencia en revisión de documentos.

Almacenamiento y recuperación de documentos controlados

- Los requisitos exigidos para el almacenamiento y la recuperación de documentos tratan de asegurar la seguridad y la disponibilidad continuada.
- Estos requisitos se aplican tanto a copias físicas como electrónicas.
- Veremos 3 tipos de requisitos:
  - El propio almacenamiento.
  - La circulación y el acceso a los documentos.
  - La seguridad, incluyendo el borrado.

Almacenamiento y recuperación de documentos controlados

- El almacenamiento debe indicar:
  - El número de copias almacenadas.
  - El equipo responsable de almacenar cada copia.
- El modo de almacenamiento. Obviamente, el formato electrónico es más económico y requiere menos espacio.
- En ocasiones es necesaria la copia en papel por motivos legales, por lo que se guarda esta y una copia electrónica.

Almacenamiento y recuperación de documentos controlados

- La circulación y el acceso requieren:
  - Instrucciones para enviar un documento a tiempo a unos determinados destinatarios.
  - Acceso a las copias de forma eficiente y sencilla, cumpliendo con las restricciones de seguridad.
- Como en el caso anterior, estos requisitos se aplican tanto a envíos de papeles como a correo electrónico, intranet e Internet.

Almacenamiento y recuperación de documentos controlados

- La seguridad requiere:
  - Restringir el acceso a ciertos documentos.
  - Impedir cambios no autorizados en los documentos almacenados.
  - Copias de seguridad, tanto físicas como electrónicas.
  - Determinar el periodo de almacenamiento.
- Al finalizar el periodo de almacenamiento, los documentos se pueden destruir o almacenar en otros dispositivos secundarios con un acceso más limitado.
- Es necesario hacer notar que las copias físicas pueden sufrir daños por incendios o inundaciones, mientras que los discos duros pueden tener problemas electrónicos.
- Las copias de seguridad deben tener en cuenta estas debilidades, así como la importancia de los datos.

Contenidos

Control de progreso del proyecto

- Los retrasos en los plazos o los desvíos en el presupuesto que superan el 10 % son situaciones que llevan a graves problemas.
- Estas situaciones suelen deberse a:
  - Unos plazos/presupuestos demasiado optimistas.
  - Riesgos mal planificados o mal manejados.
  - Incapacidad de detectar antes los problemas, y por tanto ser incapaz de tomar medidas.
- La primera situación se debe prevenir usando revisiones y herramientas de planificación.
- La segunda y la tercera situación deben tratarse con el control del progreso del proyecto.

Control de progreso del proyecto

- Mientras que las revisiones se centran más en los aspectos técnicos del proyecto, el control de progreso se centra en la administración, tanto de recursos humanos como de plazos y presupuestos.
- La importancia de usar garantía de calidad en esta fase se observa estadísticamente al comprobar que los proyectos software se retrasan en general más que, por ejemplo, la ingeniería civil.
- Ya vimos en la introducción que este hecho guarda relación con los aspectos particulares del software respecto a los proyectos "físicos".
- Prestaremos especial atención a las dificultades que suponen los colaboradores externos y los proyectos internos.

Componentes del control de progreso

- El control de progreso tiene un objetivo fundamental: detectar las irregularidades tan pronto como sea posible.
- Esta detección permite iniciar las respuestas cuanto antes.
- La información obtenida del control de progreso, sus éxitos y fracasos, sirven para un objetivo a largo plazo: las acciones preventivas.
- Los componentes del control de progreso son:
  - Control de riesgos.
  - Control de los plazos.
  - Control de los recursos.
  - Control del presupuesto.

Control de riesgos

- En este componente nos referimos a los riesgos identificados en el ante-proyecto, a los identificados en la revisión del proyecto y en el plan de proyecto, y a todos aquellos riesgos identificados durante el proyecto.
- El equipo de desarrollo debe aplicar actividades de control del riesgo a cada riesgo que realmente aparece.
- El control de los riesgos comienza con la preparación de evaluaciones periódicas sobre los riesgos aparecidos y los posibles resultados de las acciones que se están llevando a cabo para solucionarlos.
- Basándose en estas evaluaciones, los encargados del proyecto pueden intervenir y ayudar a encontrar una solución.
- Existen varios estándares y libros dedicados a los riesgos en los proyectos.

Control de plazos

- Este componente trata de adecuar los plazos a las fechas aprobadas en el contrato.
- El seguimiento se basa en hitos, que se ponen para facilitar la identificación de los retrasos.
- Los hitos que suponen la entrega de ciertos productos al cliente son de especial importancia.
- Aunque se pueden anticipar algunos retrasos, el control se debe concentrar en los retrasos críticos, es decir, aquellos que pueden afectar la fecha final del proyecto.
- La mayoría de la información de control de plazo se transmite con los informes de hitos y con informes periódicos.
- Al recibir estos informes, los jefes de proyecto pueden intervenir para dedicar más recursos o negociar una nueva fecha de entrega con el cliente.

Control de recursos

- Este componente se centra en los recursos humanos pero también puede referirse a otros elementos.
- Es importante relacionarlo con los retrasos en el desarrollo, porque no es lo mismo haber consumido ciertos recursos hasta la fecha y tener las tareas terminadas que haberlos consumido sin haberlas acabado.
- Por ejemplo, cuando tenemos sistemas en tiempo real suelen ser necesarios recursos extra para el desarrollo y el testing.
- También en este caso el control se basa en informes periódicos de los recursos en comparación con los planificados.
- Es importante notar que pequeños cambios en fases tempranas del proyecto pueden resultar en grandes cambios en las siguientes fases del desarrollo.
- Por ejemplo, un desvío del 5 % en el diseño puede suponer un desvío mayor del 25 % al acabar la implementación.

Control de recursos

- Otro aspecto del control de recursos es la composición interna o asignación.
- Por ejemplo, es posible que veamos que, en un cierto proyecto, la implementación se ha terminado usando tantos analistas como habíamos previsto.
- Sin embargo, si se mira el tipo de analistas resulta que, en lugar de haber utilizado un 20 % de analistas senior, hemos usado un 50 %.
- Este problema "interno" supondrá problemas presupuestarios en el futuro.
- Aunque parezca que este punto debería tratarse en los controles del presupuesto (y de hecho se hará), es interesante detectarlo en este punto, ya que es anterior.
- Si los cambios están justificados, se puede administrar para tener en cuenta los recursos extra.
- También es posible reorganizar los equipos, cambiar el plan de proyecto, o tomar otras medidas.

Control del presupuesto

- Este componente se ocupa de comparar los costes presupuestados con los que realmente se han necesitado.
- Como en el caso anterior, es importante tener en cuenta los retrasos en el desarrollo.
- Los principales elementos que necesitan control del presupuesto son:
  - Recursos humanos.
  - Elementos para desarrollo y testing.
  - Compra de software.
  - Compra de hardware.
  - Pagos a subcontratas.
- De nuevo, el control de los recursos se basa en hitos y en informes periódicos.

Control del presupuesto

- Cuando nos encontramos con desviaciones debidas a causas internas, las medidas son similares al caso del control de recursos.
- Si las causas son externas, se pueden tomar otras medidas, incluyendo medidas legales.
- El control del presupuesto es de vital importancia, pues afecta directamente a la viabilidad del proyecto.
- Por ello, a menudo este es el único elemento del que se hace seguimiento.
- Esto es contraproducente, porque controlar el resto de aspectos finalmente servirá para controlar el presupuesto, pues los retrasos y la falta de recursos suponen costes extra.
- Por ello, a largo plazo es más barato controlar todos los componentes.

Control en proyectos internos y subcontratados

- El control de progreso se inicia para poder administrar el proyecto teniendo en cuenta el *estado general* del desarrollo.
- Sin embargo, en muchos casos el control de proyecto proporciona únicamente una visión limitada del progreso de los proyectos internos, e incluso una visión más limitada del progreso de los participantes externos.
- Es necesario evitar el "descontrol" de este tipo de proyectos.
- Los proyectos internos, normalmente encargados a algunos departamentos específicos, no suelen disponer de clientes externos.
- Por ello, se les suele dedicar poca atención desde el punto de vista de la administración.
- Además esto se puede ver aumentado por la falta de seguimiento por parte del "cliente interno".
- Esto lleva a identificar tardíamente los retrasos.
- Es por tanto inevitable, para evitar estos problemas, imponer un amplio rango de controles a los proyectos internos.

Control en proyectos internos y subcontratados

- Entre los participantes externos se incluyen las subcontratas, los proveedores de software, el software reutilizado y, en algunos casos, incluso al cliente.
- Cuanto mayor y más complejo es el proyecto, más probable es que los agentes externos sean necesarios, y más trabajo les sea asignado.
- Las subcontratas no son casos excepcionales, casi todos los proyectos las necesitan, principalente, por falta de personal o porque son más baratas.
- Muchas veces los términos del contrato se complican tanto que la comunicación y coordinación se vuelven muy problemáticas.
- Son necesarios mayores esfuerzos para llegar a un cierto nivel de control.
- Por tanto, el control de progreso para externos debe centrarse en los plazos y en los riesgos identificados en el plan de desarrollo.

Control en proyectos internos y subcontratados

Observación

- *A lo largo del tema hemos visto que muchas indicaciones de las que damos aquí son de "sentido común", y por lo tanto resulta extraño que no se apliquen o que sea necesario hacerlas explícitas.*
- *Sin embargo, en los últimos puntos vemos un claro ejemplo de intereses económicos que chocan con el sentido común.*
- *Las empresas subcontratadas quieren evitar controles porque quieren recortar costes, así que deben establecer ciertas cláusulas que eviten excesiva interferencia del cliente.*
- *Por tanto, no podemos aplicar los controles que el software probablemente necesite, pero a nosotros también nos resulta beneficioso tener una empresa que nos haga el trabajo de forma "barata".*
- *En conclusión, esta es una de las razones por las que el software acaba fallando más que los puentes.*

Implementación de regímenes de control de progreso

- El control de progreso se suele basar en procedimientos que determinan:
  - La asignación de responsabilidad de las tareas de control que son más apropiadas para el proyecto, incluyendo:
    - Las personas a cargo del control de progreso.
    - La frecuencia de los informes de cada unidad.
    - Las situaciones en que los jefes de proyecto deben comunicarse con los encargados de administración.
    - Las situaciones en que cargos intermedios deben informar a sus superiores.
  - Administración de los informes de control, incluyendo:
    - Si los informes se están transmitiendo adecuadamente.
    - Las actividades concretas de control que se deben iniciar.

Implementación de regímenes de control de progreso

- En organizaciones grandes, el control de progreso puede estar estructurado en varios niveles.
- Por ejemplo, a nivel de equipo, a nivel de departamento y el nivel más alto, de coordinación.
- Aunque a cada nivel se establecen diferentes protocolos para el control del progreso, es necesario que se incluya toda la información necesaria para poder coordinarse.
- La cadena de informes transmite la información recogida de todos los niveles inferiores.
- El informe final, del líder de proyecto, resume el estado de los riesgos, plazos, recursos y presupuesto.
- El líder del proyecto basa sus informes de progreso en la información recogida de los líderes de equipo.

Implementación de regímenes de control de progreso

Project Leader's Progress Report For the period: _____

The project: _____

1 Status of software risks

No.	Risk item	Activities involved	Other projects involved	Solved	Risk severity	Comments
1						
2						
3						
4						
5						
6						

Risk severity: 1 – Solution expected within one month. 2 – Solution expected within 3 months. 3 – Solution expected within 6 months. 4 – Solution directions are available, good success prospects. 5 – All trials failed, no possible solution is identified.

2 Status of resources use

No.	Activity	Hours Worked				Percent of activity completed	Comments
		Planned	Used prior to report period	Invested during report period	Total invested		
1							
2							
3							
4							

Implementación de regímenes de control de progreso

3 Project completion estimates (mark the most probable estimate)

Human resources	Completed with less than planned	No additional resources required	10% Excess	20% Excess	30% Excess	40% Excess	50% Excess or more
Timetable	Completed before planned date	Completed on time	2 weeks delay	1 month delay	2 months delay	4 months delay	6 months delay and more

Comments:

Signed: Name:_____ Date: _____ Signature:_____

Software para el control del proyecto

- No se puede esperar llevar el control de proyecto sin ayuda de algún tipo específico de software.
- Desde hace años se dispone de programas que permiten un seguimiento integral de los proyectos.
- La mayoría de estas herramientas generan diagramas PERT que permiten ver la información por separado para cada actividad.
- Suelen ser también muy configurables, de manera que se adaptan bien a necesidades específicas.
- A continuación se muestran algunos ejemplos de elementos para los que es interesante este tipo de herramienta.
- Para el control de los riesgos:
  - Listas de riesgos categorizadas y la fechas estimadas de corrección.
  - Listas con los riesgos software críticos, y fechas límite a partir de las cuales el proyecto se retrasa.

Software para el control del proyecto

- Para el control de los plazos:
  - Listas clasificadas de actividades retrasadas.
  - Listas clasificadas de actividades *críticas* retrasadas.
  - Plazos actualizados con respecto a los informes y a las medidas de corrección aplicadas.
  - Listas clasificadas de hitos retrasados.
  - Fechas actualizadas para los hitos retrasados.
- Para el control de recursos:
  - Plan de asignación de recursos por actividad, módulos, equipos, periodos, etc.
  - Uso de recursos, por periodo y acumulado.
  - Uso de recursos *críticos*, por periodo y acumulado.
  - Asignación de recursos actualizada de acuerdo a los informes y las medidas de corrección.
- Para el control de presupuestos:
  - Presupuestos por actividad y módulo, distinguiendo equipos, periodos, etc.
  - Informes de uso del presupuesto, por periodo o acumulado.
  - Informes de desvío del presupuesto, por periodo o acumulado.
  - Presupuestos actualizados con las medidas de corrección necesarias.

Contenidos

Métricas

- La idea de las métricas es asignar un "número" a las actividades de desarrollo de software para medir "cómo de bien" se están llevando.
- Tenemos 2 definiciones para las métricas:
  - Una medida cuantitativa de la medida en la que un elemento posee un cierto atributo de calidad.
  - Una función cuya entrada son datos software y cuya salida es un valor numérico que se puede interpretar como la cantidad de calidad que posee.
- Es especialmente interesante añadir métricas para:
  - Medir la calidad durante desarrollo y el mantenimiento.
  - Ayudar a tomar decisiones.
  - Iniciar acciones correctivas.
- El análisis estadístico de métricas debe predecir los cambios a resultas del uso de nuevas herramientas, procedimientos, etc.

Métricas

- El campo de uso de las métricas se ha ampliado mucho durante las últimas décadas.
- Sin embargo, la relación entre métricas y garantía de calidad no se ha desarrollado demasiado.
- Una de las razones de esta falta de uso es que no se han obtenido los resultados que en principio se esperaban.
- Por ello, solo una pequeña parte del proceso de desarrollo aplica métricas de manera sistemática.
- En esta sección nos centraremos más en los aspectos generales que en métricas concretas, ya que cada empresa usa unas concretas, y las más generales son fácilmente accesibles.

Los objetivos de medir la calidad

1. Facilitar la administración. Este objetivo se base en calcular las métricas sobre:
   - Los desvíos de rendimiento.
   - Los desvíos de los plazos y del presupuesto.
2. Identificar las situaciones que requieren mejoras. Este objetivo se basa en:
   - La acumulación de métricas con información sobre el rendimiento de los equipos, las unidades, etc.

Los objetivos de medir la calidad

- Las métricas se basan en medir indicadores, valores cuantitativos como:
  - Adecuación a los estándares.
  - Los objetivos de calidad fijados para las personas.
  - Los objetivos de calidad alcanzados el año anterior.
  - Los objetivos de calidad alcanzados en el proyecto anterior.
  - Los objetivos de calidad alcanzados por otros grupos usando el mismo proceso de desarrollo.
  - La media de objetivos de calidad alcanzados por la empresa.

Los objetivos de medir la calidad

- Para que una métrica sea útil, debe cumplir unos ciertos requisitos:
  - Ser relevante, es decir, medir un atributo de importancia.
  - Ser válida, es decir, medir realmente el atributo indicado.
  - Ser confiable, es decir, devuelve resultados parecidos si se ejecuta varias veces en condiciones similares.
  - Debe ser general, es decir, aplicable en distintas implementaciones.
  - Debe ser exclusiva, es decir, no medir distintos atributos.
  - Debe ser simple, es decir, sencilla (y barata) de implementar.
  - Debe ser autónoma, es decir, no depender de otros datos.
  - Debe ser objetiva, es decir, inmune a usos interesados.

Clasificación de métricas

- Hay dos clasificaciones para las métricas.
- Según la fase en la que se necesitan:
  - Métricas del proceso, usadas durante el desarrollo.
  - Métricas del producto, usadas durante el mantenimiento.
- Según qué miden:
  - Calidad.
  - Plazos.
  - Eficiencia, bien de eliminar defectos o de mantenimiento.
  - Productividad.
- Hay dos medidas especialmente interesantes que conviene conocer:

  KLOC, que indica las líneas de código (por miles).

  Puntos de función, que mide los recursos humanos necesarios para desarrollar un programa. Esta medida se calcula con un algoritmo complejo que no veremos, pero la idea básica es que, cuanto más puntos de función tengamos, más compleja es la tarea.

Métricas del proceso

- Las métricas del proceso a cargo de la calidad básicamente definen fórmulas para relacionar la cantidad de errores y su severidad.
- Las métricas del proceso a cargo de los plazos miden:
  - La relación entre hitos alcanzados en plazo con respecto a aquellos en los que se falló.
  - Retraso medio en los hitos.
- Las métricas del proceso a cargo de la eficiencia relacionan los errores encontrados durante el desarrollo con los encontrados por el cliente durante un cierto periodo (e.g. un año).
- Las métricas del proceso a cargo de la productividad usan la cantidad de personal utilizado y el nivel de reutilización de software.

Métricas del producto

- Nos centraremos en el mantenimiento correctivo.
- En este caso las métricas del producto distinguen dos tipos de servicio:
  - El servicio técnico, es decir, la ayuda que requiere el usuario para hacer funcionar la aplicación.
  - Los servicios correctivos, es decir, las correcciones que es necesario hacer en el producto debido a errores.
- Normalmente un solo centro se encarga de ambos servicios.
- En general se atenderán consultas, y solo en ciertos casos se recibirán errores.
- Estos errores se reenviarán al departamento correspondiente con un informe.
- Por tanto, las métricas de servicio técnico incluyen a las de servicios correctivos.
- Se suelen utilizar datos acumulados durante un periodo largo, al menos un año.

Métricas del producto

- Las métricas de servicio técnico a cargo de la calidad incluyen:
  - La densidad de consultas, es decir, la cantidad de consultas recibidas.
  - La severidad de las consultas.
  - El éxito de las respuestas, es decir, la cantidad de consultas que han satisfecho la consulta hecha por el usuario.
- Las métricas de servicio técnico a cargo de los plazos no se aplican.
- Las métricas de servicio técnico a cargo de la eficiencia miden los recursos necesarios con respecto al número de consultas.
- Las métricas de servicio técnico a cargo de la productividad relacionan las líneas de código con el número de consultas.

Métricas del producto

- En las métricas de servicios correctivos a cargo de la calidad podemos distinguir 2 casos:
  - Errores del sistema, que deben arreglar los encargados del mantenimiento.
  - Errores de los encargados del mantenimiento al arreglar los errores anteriores.
- Teniendo esto en cuenta, tenemos las siguientes métricas:

  Densidad de fallos del sistema. Compara los errores identificados por el usuario con los detectados durante el desarrollo.

  Severidad de los fallos del sistema. Estudia la severidad de los fallos.

  Fallos del mantenimiento. Estudia los errores del equipo de mantenimiento, lo que incluye las correcciones fuera de plazo.

  Disponibilidad. Estudia el tiempo que los servicios de corrección están disponibles para el usuario.

Métricas del producto

- Las métricas de servicios correctivos a cargo de los plazos no se aplican.
- Las métricas de servicios correctivos a cargo de la eficiencia estudian los recursos necesarios para arreglar un fallo.
- Las métricas de servicios correctivos a cargo de la productividad estudian los recursos totales necesarios para mantener el sistema.

Además de las métricas estándar

- En los últimos años han aparecido nuevas métricas, muchas asociadas al proceso.
- Algunas han tenido éxito y han sido adoptadas por algunas empresas.
- Un ejemplo de esta tendencia es el test de Joel.
- El test consta de 12 preguntas sencillas que permiten medir la madurez del proceso de desarrollo.
- Estas preguntas son:
  1. ¿Se usa software de control de versiones?
  2. ¿Es posible construir el sistema completo en un solo paso?
  3. ¿Se construye el sistema completo todos los días?
  4. ¿Hay una base de datos de *bugs*?

Además de las métricas estándar

5. ¿Se arreglan los *bugs* antes de escribir nuevo código?
6. ¿Se actualiza la planificación?
7. ¿Hay una especificación?
8. ¿Tienen los programadores un entorno tranquilo?
9. ¿Tenéis las mejores herramientas que el dinero puede comprar?
10. ¿Hay encargados de testing?
11. ¿Se hace programar a los candidatos en la entrevista de trabajo?
12. ¿Hacéis "pruebas de pasillo de usabilidad" (*hallway usability testing*)?

- Una puntuación de 10 o menos en el test se considera mala.
- Más información sobre esta métrica está disponible aquí.

Implementación de métricas de calidad

- Para usar métricas en una empresa es necesario:
  - Definir las métricas relevantes para cada equipo, departamento, etc.
  - Aplicar las métricas regularmente.
  - Calcular estadísticas.
  - Se deben usar para:
    - Cambiar el desarrollo, el mantenimiento y cualquier otra fase en la que se recojan métricas.
    - Cambiar las métricas, si no son útiles.
    - Planear acciones correctivas.
- Es decir, no basta con calcular métricas, es necesario darles un buen uso y adaptarlas a las circunstancias.

Definición de nuevas métricas

- Para definir nuevas métricas se deben seguir 4 pasos:
  1. Definir los atributos que se van a medir: calidad, productividad, etc.
  2. Definir las métricas que medirán esos atributos y confirmar su adecuación.
  3. Determinar los valores con los que se compararán: los valores obtenidos el año pasado, otro grupo, etc. Estos valores sirven de indicadores para saber si el atributo reúne ciertas características.
  4. Determinar los métodos de aplicación:
     - Método de comunicación, incluyendo el tipo de informes y su frecuencia.
     - Método de recolección de métricas.
- Las nuevas métricas pasarán a usarse igual que el resto de métricas.

Definición de nuevas métricas

Discusión

¿Cómo definirías una métrica para medir tu rendimiento académico?

Limitaciones de las métricas

- Hay 3 tipos de obstáculos con los que se pueden encontrar las métricas:
  - Problemas presupuestarios para asignar recursos a calcular y aplicar métricas.
  - Factores humanos, en especial oposición de los empleados a las comparaciones.
  - Dudas sobre la validez de los datos, que pueden haber sido manipulados (aunque, como hemos visto, métricas que no sean objetivas no deberían usarse).
- Estas dificultades son bastante generales y no se aplican solo al software.
- Sin embargo, también existen algunos problemas específicos.

Limitaciones de las métricas

- Los mayores problemas específicos al desarrollo a los que se enfrenten las métricas del software son:
  1. El estilo de programación afecta a una de las medidas más básicas: el número de líneas de código.
  2. Además, esta medida también se ve afectada por la cantidad de comentarios en el código.
  3. No "cuesta" lo mismo una línea de código para un método muy complicado que para uno sencillo.
  4. Cuanto más código se reutilice, más código se termina por día sin que eso suponga el mismo esfuerzo que si se programa el programa entero.

Limitaciones de las métricas

5. La profesionalidad de los equipos de revisión y testing, que pueden detectar más defectos en el mismo tiempo, y por tanto reducir mucho los costes.

Observación

Fijaos que es difícil juzgar el trabajo de estos equipos, porque es complicado saber si los errores que posteriormente encuentra el usuario (y, como hemos visto, lo normal es que siempre acabe encontrando alguno) eran detectables por los equipos de revisión. ¿Trabajaron realmente duro o "vaguearon" y dejaron pasar más defectos de la cuenta?

6. El estilo de los informes. Si el estilo de un informe es muy conciso puede parecer que se encontraron menos errores que si el estilo es más detallado.

Limitaciones de las métricas

- En cuanto a los problemas con las métricas durante el mantenimiento, tenemos:
  1. La calidad en el mantenimiento depende del desarrollo: un mal desarrollo supone muchas consultas y, por tanto, malos resultados en el mantenimiento.
  2. Como en el caso del desarrollo, las líneas de comentarios pueden incrementar mucho las líneas de código necesario en el mantenimiento, falseando resultados.
  3. Los módulos complejos necesitan más mantenimiento que los simples, aunque tengan las mismas líneas de código.
  4. Si se reutiliza mucho código, es de esperar que no sea necesario tanto mantenimiento, aunque esto no significa que el mantenimiento sea mejor.
  5. La cantidad de usuarios. Si hay muchos usuarios es normal tener muchas consultas, aunque haya pocos errores (en general habrá muchas repeticiones).

Contenidos

Costes de la calidad

- Para finalizar el tema de calidad, vamos a ver cómo estudiar el coste de la garantía de calidad como una métrica más.
- Esta métrica también se usará para ayudar a tomar decisiones sobre el proyecto.
- Además, este tipo de métricas son muy importantes porque dan una visión más general que las anteriores.
- Esta visión general ha hecho que en la actualidad sea objeto de numerosos estudios.
- Dichos estudios tratan de prevenir los efectos desastrosos que han tenido ciertos defectos en proyectos software.

Costes de la calidad

Samsung se hunde en Bolsa y dispara a sus rivales

Compartido 0

8 Comentarios

El Galaxy Note 7 y Samsung fuera del mercado

· Samsung advierte a los usuarios del Galaxy Note 7: "Deben apagarlo y dejar de usar el dispositivo"

http://www.elmundo.es/economia/2016/10/11/57fcd94346163fd03d8b458a.html

Costes de la calidad

90 euros de regalo para los que devolvieron su Samsung Galaxy Note 7 en España

Publicado por Jordi Giménez el 29 de noviembre de 2016 a las 17:00. Sin comentarios

Gestión anuncios

Samsung Galaxy note 8 | Hotel Espana

Como reservar en un hotel | Movil Espana

Comparte Twitea

https://www.actualidadgadget.com/90-euros-regalo-los-devolvieron-samsung-galaxy-note-7-espana/

Costes de la calidad

Samsung ganó un 21% menos en España en 2016 lastrado por el Note 7

La filial del gigante coreano acusó también el abandono de la gama baja de móviles

RAMÓN MUÑOZ

Madrid - 11 SEP 2017 - 21:07 CEST

Usuarios prueban las gafas de realidad virtual de Samsung. FABRIZIO BENSCH (REUTERS)

MASTERS • OPOSICIONES
SEMINARIOS • CURSOS

Máster en
ASESORÍA DE EMPRESAS

Cumpliendo 40 años

https://economia.elpais.com/economia/2017/09/11/actualidad/1505127980_628300.html

Objetivos de las métricas de coste de calidad

- El uso de métricas de coste de calidad permite a los administradores tener el control económico de las actividades de calidad y de sus resultados.
- Los objetivos concretos son:
  - Controlar los gastos iniciados por la propia organización para prevenir y detectar defectos software.
  - Evaluar los daños económicos de los fallos software para decidir si revisar el presupuesto dedicado a garantía de calidad.
  - Evaluar los planes para incrementar o decrementar las actividades de garantía de calidad o para invertir en un nuevo plan de calidad basándose en los datos almacenados.

Objetivos de las métricas de coste de calidad

- El control administrativo del coste de la calidad compara las métricas con:
  - El gasto previsto.
  - Los costes de los fallos durante el año anterior.
  - Costes de calidad del proyecto anterior.
  - Los costes de error y control de calidad en otros departamentos.
- Cuando se modifica el plan de calidad, se pueden usar los siguientes factores para estudiar su éxito.
  - Porcentaje del coste de calidad respecto al coste total de desarrollo.
  - Porcentaje de costes de fallos software respecto al coste total de desarrollo.
  - Porcentaje de coste de calidad respecto al coste total de mantenimiento
  - Porcentaje de coste de calidad del total de las ventas de productos software y de servicios de mantenimiento.

El modelo clásico de coste de calidad

- El modelo clásico clasifica los costes en:
  - Costes de control, que incluyen los costes para prevenir y detectar errores.
  - Costes de fallo de control, que incluyen los fallos que ocurrieron al no ser capaces de prevenir o detectar ciertos errores.
- Los costes de control se dividen en:
  - Costes de prevención. Se refieren a las inversiones generales (no dirigidas a un proyecto o sistema concreto) en calidad.
  - Costes de evaluación. Incluyen los costes de las actividades para detectar errores en un proyecto concreto.
- Los costes de fallo de control se dividen en:
  - Costes de fallo interno. Se refieren a los errores detectados por personal interno.
  - Costes de fallo externo. Se refieren a los errores detectados por el cliente.

Costes de prevención

- Los típicos costes de prevención incluyen:
  1. Inversiones para desarrollar o mejorar la infraestructura de calidad, o para actualizarla regularmente.
     - Creación de procedimientos e instrucciones de trabajo.
     - Métricas de calidad.
  2. Implementar regularmente actividades preventivas de calidad:
     - Formar a los empleados en nuevas tecnologías y procedimientos.
     - Exigir a los empleados la certificación necesarias para llevar a cabo ciertas tareas.
     - Hacer consultas periódicas a los líderes de equipo sobre la calidad.
  3. Control del rendimiento del sistema de calidad mediante:
     - Revisiones internas de calidad.
     - Auditorías externas por organizaciones de certificación de calidad.
     - Revisiones de la administración de la calidad.

Costes de evaluación

- Los típicos costes de evaluación son:
  1. Revisiones:
     - Revisiones formales del diseño.
     - Revisiones por pares.
     - Revisiones por expertos.
  2. Costes de testing.
  3. Costes para asegurar la calidad de terceros:
     - Subcontratas.
     - Proveedores de software específico y del código reutilizado.
     - Control de los módulos proporcionados por el cliente.

Costes de fallo interno

- Como hemos visto, los fallos internos son aquellos que se encuentran en las revisiones y en el testing antes de haberse enviado al cliente.
- En resumidas cuentas, estos costes son simplemente los resultantes de arreglar estos errores.
- En algunos casos si es alguien del equipo quien encuentra algún error no se consideran costes internos, sino de desarrollo, porque la revisión era informal.
- Los costes típicos por fallo interno son:
  - Costes de re-diseño.
  - Costes de re-implementación.
  - Costes de repetición de la revisión de diseño y de re-testing.
- **Nota:** según el motivo de la revisión y el testing, se pueden considerar costes de evaluación o de fallo interno.

Costes de fallo externo

- Los costes de fallo externo son aquellos que resultan de las correcciones necesarias una vez se ha instalado el producto en el cliente.
- Estos costes pueden directos o indirectos.
- En general, los costes de fallo externo se refieren a los fallos directos, que incluyen:
  - Resolución de quejas durante el periodo de garantía. En muchos casos esto supone una revisión de la queja y la transmisión de instrucciones. Es frecuente que las quejas se deban a problemas con el manual de instrucciones.
  - Corrección de bugs detectados por el cliente. Esto supone no solo corregir el programa, también sustituir la versión del cliente, quizás con gastos de desplazamiento.
  - Corrección de errores después del periodo de garantía.
  - Devolución del importe en caso de insatisfacción.
  - Indemnización en caso de fallos graves.

Costes de fallo externo

- Los costes indirectos suelen ser mucho mayores que los directos.
- Además son mucho más complicados de estimar.
- Son consecuencia de los fallos directos.
- Cuanto mayor sea la cantidad de fallos directos, más se verán afectados los costes indirectos.
- Típicos casos de costes externos indirectos son:
  - Reducción de ventas por insatisfacción con productos anteriores.
  - Pérdida de credibilidad de la marca.
  - Mayores gastos en publicidad.
  - Necesidad de reducir los precios para captar clientes.

El modelo extendido de coste de calidad

- El modelo clásico es bastante general, y no tiene en cuenta ciertos aspectos que suponen grandes costes.
- En especial, no incluye los gastos que provienen de una mala administración de los plazos, el presupuesto o el personal.
- El modelo extendido añade los costes por fallo de administración.

Costes por fallo de administración

- Los errores por fallo de administración se pueden dar en cualquier momento del desarrollo
- Los más corrientes se deben a fallos al estimar los plazos y los presupuestos.
- También se deben a no detectar estos errores a tiempo.
- Los costes por fallo de administración incluyen:
  - Costes no planeados, como resultado de subestimar los recursos necesarios.
  - Pagos a clientes en compensación por completar tarde un proyecto, debido a plazos o presupuestos poco realistas.
  - Pagos a clientes en compensación por completar tarde un proyecto, debido a no contratar al equipo adecuado.
  - *Efecto dominó:* daños causados a otros proyectos desarrollados por los mismos equipos. Si un equipo debe dedicar tiempo extra a arreglar algo en un proyecto, no puede continuar con el desarrollo del resto. Esto es un fallo de administración porque se deberían haber asignado estos proyectos a equipos diferentes. También podemos sufrir un efecto dominó de costes indirectos.

Problemas para aplicar los costes de calidad

- El cálculo de los costes de calidad es un problema general de la ingeniería.
- Esto se debe a la falta de complección o exactitud de los datos debida a:
  - Una identificación incompleta o inexacta de los costes que queremos calcular.
  - Informes incorrectos.
  - Informes parciales, en muchos casos para ocultar costes internos y externos.
  - Información parcial de costes por fallo externo, ya que se ocultan ciertas compensaciones al cliente como descuentos futuros, atención al cliente gratuita, etc.

Problemas para aplicar los costes de calidad

- Además, están los problemas al calcular costes desde el punto de vista administrativo:
  - Las actividades de revisión del contrato y de control de progreso a menudo se hacen "a tiempo parcial", y además están divididas en muchas otras actividades breves. Por ello, no es raro que alguna de estas sub-actividades sencillamente no se realice.
  - Algunos de los participantes en estas tareas son senior que no tienen que notificar el tiempo que dedican.
  - Es difícil asignar la responsabilidad cuando hay errores de plazo. Es posible que sea el cliente, al haber especificado mal los requisitos, el equipo de programación o los encargados de la administración.
  - Cuando es necesario compensar al cliente, este pago no se hace inmediatamente, sino que requiere mucho tiempo (si es necesario un juicio puede llevar años). Por ello, es difícil clasificarlo y usarlo en otros proyectos.

Calidad del software

Calidad: conclusiones

- En este tema hemos visto varias cosas sobre calidad:
  - Las factores que influyen en la calidad.
  - Cómo conseguir calidad al desarrollar software.
  - Cómo hacer el seguimiento de la calidad.
  - Cómo medir la calidad.
- Muchas de las ideas de este tema son de "sentido común", pero es necesario conocerlas para poderlas aplicar.
- Problablemente sea necesario adaptar algunos conceptos según la empresa, el tipo de proyecto y el personal disponible.
- Sin embargo, es necesario tenerlas todas en cuenta para poder tener un producto final de calidad.