¿Es mejorable el proceso software seguido?

Material bibliográfico

- "Ingeniería del software. Un enfoque práctico". Roger S. Pressman. 7ª edición. McGraw-Hill.
- "Calidad de sistemas informáticos". Mario G. Piattini Velthius, Félix O. García Rubio e Ismael Caballero Muñoz-Reja. Ra-Ma.
- "La calidad del software y su medida". Jesús Mª Minguet Melián y Juan F. Hernández Ballesteros. Editorial Centro de Estudios Ramón Areces.
- Software Engineering Institute (SEI): http://www.sei.cmu.edu/.
- European Software Institute (ESI-Tecnalia): http://www.tecnalia.com/es/.

Índice

- Objetivamente, ¿es mejorable el proceso software empleado?
- ¿Cómo es posible mejorarlo?

Objetivamente, ¿es mejorable el proceso software empleado?

Respuesta

- Para responder, es necesario saber:
  - Meta que se quiere alcanzar.
  - Cómo de cerca se está de dicha meta (si se cumple con ella o no).
- A continuación se consideran ambos aspectos, teniendo en cuenta un proyecto software:
  - Qué objetivos se persiguen.
  - Su grado de cumplimiento.

Objetivos (I)

- En general, en todo nuevo proyecto software se persiguen los siguientes objetivos:
  - Con respecto a la gestión del proyecto:
    - Cumplir con el esfuerzo estimado.
    - Cumplir con el tiempo estimado.
    - Cumplir con el coste estimado.
  - Con respecto al producto propiamente dicho:
    - Lograr una alta calidad.

Objetivos (II)

- Para los anteriores aspectos es necesario establecer sus medidas correspondientes para poder cuantificar y comparar:
  - Esfuerzo: horas*hombre, días*hombre, …
  - Tiempo: días, meses, …
  - Coste: €, $, …
- Pero …
  - ¿Y la calidad del software?
  - ¿Cómo se puede medir?

Objetivos (III)

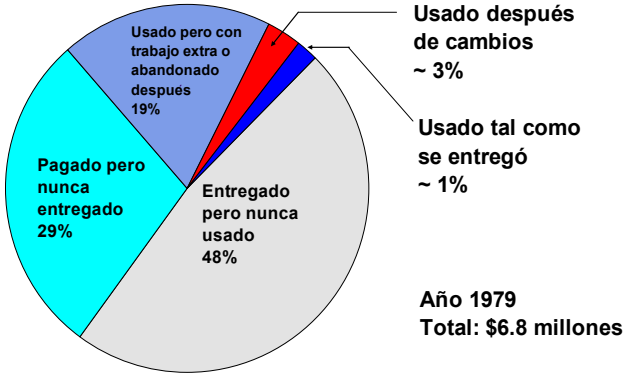
- Hay muchas definiciones de calidad, cada una enfatizando un aspecto concreto.
  - J. M. Juran, uno de los grandes autores en temas de calidad, recopiló hasta 13 significados distintos del término calidad como un ejemplo de las múltiples acepciones que se pueden encontrar.
- Por ello, para cada empresa y, o, producto es necesario establecer lo que se entiende por calidad, haciendo énfasis es un aspecto u otro según sus necesidades, objetivos, clientes, productos, actividades, …

Objetivos (IV)

- Sólo se imponen las siguientes limitaciones:
  - Establecer por anticipado los parámetros que midan la calidad desde el punto de vista (acepción) escogido.
  - Dichos parámetros deben ser:
    - Cuantificables (deben poder medirse).
    - Verificables (debe existir un procedimiento objetivo y establecido para verificar su valor).
- Ejemplos de parámetros de calidad en software:
  - Satisfacción del usuario con la funcionalidad y la facilidad de uso.
  - Número de errores incluidos.
  - Tiempo medio transcurrido entre fallos.

Grado de cumplimiento (I)

- A finales de los 70 (Crisis del Software):
  - Proyectos software contratados por el DoD Americano:



Grado de cumplimiento (II)

- A finales de los 80:
- Capers Jones estudia el software adquirido por la Administración Pública Americana:
  - Sólo entre el 5% y el 10% era directamente usable.
  - Entre el 30% y el 40% nunca se había usado o nunca se podría usar.

Grado de cumplimiento (III)

- En la década de los 90:
- Standish Group en su Chaos 2001 Report:
  - Proyectos de desarrollo “exitosos”:
    - 16% en 1994.
    - 27% en 1996.
    - 26% en 1998.
    - 28% en 2000.

Grado de cumplimiento (IV)

- Además, en el anterior informe:
- El porcentaje de características requeridas entregadas en proyectos de desarrollo “challenged”, que fueron:
  - 53% de los proyectos en 1994.
  - 49% de los proyectos en 2000,
- fue de media sólo:
  - 61% en 1994.
  - 67% en 2000.

Grado de cumplimiento (V)

- A mediados de los 90:
- Capers Jones (en su libro Assessment & control of software risks) analiza riesgos en los que se suele caer en los proyectos software y cuántos se ven afectados:
  - Proyectos de software de gestión:
    - Surgir nuevos requisitos: 80%.
    - Baja calidad: ~70%.
    - Sobrecoste: ~60%.
    - Mala gestión de la configuración: 50%.
  - Proyectos de software de sistemas:
    - Proyectos largos: 70%.
    - Excesivo papeleo: 60%.
    - módulos propensos a errores: 50%.

Grado de cumplimiento (VI)

  - Proyectos de software comercial:
    - Inadecuada documentación de usuario: 70%.
    - Baja satisfacción de usuarios: 55%.
    - Comercialización posterior a lo previsto: 50%.
  - Proyectos software subcontratados:
    - Altos costes de mantenimiento: 60%.
    - Problemas entre cliente y subcontrata: 50%.
    - Criterios de aceptación imprevistos: 30%.
  - “Proyectos” software de los usuarios (para uso particular):
    - Errores ocultos: 65%.
    - Software inmantenible: 60%.

Grado de cumplimiento (VII)

- En el siglo XXI:
- Standish Group en su Chaos 2006 Report:
  - Proyectos de desarrollo “exitosos”:
    - 35% en 2006 (vs. 16% en 1994).
  - Proyectos “challenged”:
    - 46% en 2006 (vs. 53% en 1994).
  - Proyectos de desarrollo “completamente fallidos”:
    - 19% en 2006 (vs. 31% en 1994).
  - Conclusiones:
    - Se va mejorando progresivamente el desarrollo de software desde el primer Chaos Report en 1994:
      - “There is less chaos in software development today than there has been since The Standish Group started reporting chaos back in 1994”.

Conclusión

- A la vista de los anteriores estudios puestos como ejemplo, claramente sí es mejorable el proceso software seguido, pero …

¿Cómo es posible mejorarlo?

Solución para la mejora (I)

- Solución para mejorar la gestión del proyecto:
  - Mejor estimación (previsión) y seguimiento y control de:
    - Esfuerzo:
      - Por el propio interés del grupo de desarrollo.
    - Tiempo:
      - Plazos de entrega para el cliente.
    - Coste:
      - Factura que compense el coste realmente acometido.

Solución para la mejora (II)

- 1ª posibilidad para solucionar la calidad del software: las pruebas. Sin embargo:
  - De media, un programador suele cometer aproximadamente 100 defectos por cada KLOC (1000 LOC).
  - Lo normal es que el éxito en las pruebas (detección de errores) sea menor del 50%. Por lo tanto, a más defectos antes de las pruebas, más defectos tras ellas.
  - Además, normalmente se prescinde de las pruebas (o de ciertos niveles de pruebas) o no se articulan correctamente.

Solución para la mejora (III)

- Eso quiere decir dos cosas:
  - Hay que mejorar la forma de hacer las pruebas.
  - Pero como tras las pruebas no se conseguirá calidad salvo que a ellas llegue algo que ya tenga calidad, hay que buscar otra alternativa.
- Solución: trabajar con calidad desde el principio y a lo largo de todo el proceso de desarrollo software. No buscar la calidad sólo en el último instante, sino desde el principio (en todo el proceso), siendo las pruebas un complemento.

Solución para la mejora (IV)

- No hay una “bala de plata” y las mejoras debe centrarse en tres aspectos claramente interrelacionados (no independientes):
  - **DESARROLLO:**
    - A través de la elección de un ciclo de vida (proceso software) adecuado a cada proyecto.
    - Asignatura: Proceso Software.
  - **GESTIÓN:**
    - A través de la aplicación de aproximaciones sistemáticas para gestionar los proyectos: esfuerzo, tiempo y coste.
    - Asignaturas: Gestión de Proyectos (planificación, seguimiento y cierre) y Proyectos de Desarrollo Software (estimación y gestión de proyectos) en itinerario de Ingeniería del Software (obligatoria).
  - **CALIDAD:**
    - A través de la mejora de la calidad del proceso y del producto.
    - Asignatura: Aseguramiento de la Calidad en itinerarios de Ingeniería del Software (obligatoria) y Sistemas de Información (optativa).