Ingeniería del Software I - Práctico N◦ 1

FaMAF, Universidad Nacional de Córdoba Libro: An Integrated Approach to Software Engineering de Jalote

1. ¿Que es la Ingeniería del Software y cual es su relación con la Ciencia de la Computación?

La Ingeniería del Software consiste en la aplicación de un enfoque *sistemático*, disciplinado, y *cuantificable* al desarrollo, operación, y mantenimiento del software. Busca brindar **procesos** de desarrollo de software que conlleven un desarrollo **productivo** (en términos *monetarios* y de *tiempo*) y un producto de **calidad** (con respecto a las características del proyecto).

Se relaciona con la Ciencia de la Computación en que esta es la encargada del diseño interno dos software.

> *Enfoque sistemático:* metodología y prácticas existentes para solucionar un problema dentro de un dominio determinado. Esto permite **replicar** el proceso y da la posibilidad de **predecir** su desarrollo (independientemente del grupo de personas que lo lleva a cabo).

---

2. ¿En qué difieren la ingeniería de software de otras ingenierías más tradicionales, como la ingeniería mecánica o la ingeniería civil?

- **Fallas:** Las fallas en software son mas frecuentes (lo que hace poco confiable al software) y no son en general consecuencia del uso y el deterioro. Ocurren como consecuencia de errores (o “bugs”) introducidos durante el desarrollo, existen desde el comienzo, sólo que se manifiestan tarde.
- **Cambios en producción:** En producción se puede cambiar fácilmente (fácilmente != sin efectos secundarios no deseados) la semántica de los productos de software para corregir errores, mejorar el producto o adaptarlo a cambios en el entorno. Los productos de software suelen ser *interdependientes* entre si directa o indirectamente, al ser cambiantes muchas veces el software de un producto debe actualizarse por cambios en el entorno del que depende, lo que causa un ciclo de cambios entre productos de software.

3. Si el objetivo principal es hacer al software mantenible, liste algunas de las cosas que usted haría y algunas de las cosas que no haría durante la implementación y el testing

Haría

- Código entendible y practico (modular,...), hecho para ser leído.
- Separar en componentes las partes del proyecto y basado en ello, el código en carpetas y archivos.
- Documentar el código.
- Testear antes de subir un nuevo cambio.

No haría

- Lo contrario a lo anterior.

4. Liste algunos de los problemas que surgirán si los métodos que usted utiliza para desarrollar aplicaciones pequeñas son utilizados para desarrollar grandes aplicaciones

- **Difícil búsqueda de código** si:
  - Se utilizaron pocas y mal divididas carpetas/archivos para mucho código.
  - No se subieron commits con nombres representativos. *Consecuencias:* Esto causa una **corrección o extensibilidad mas costosa** del código.
- **Fallas en producción** si:
  - No se testea bien (completo y siempre).
  - Se depende se software externo poco confiable.
  - No se utilizo programación defensiva. *Consecuencias:* Esto causa **trabajo extra** que suele venir con **mas presión** para los trabajadores (lo que puede llevar a **descontento laboral** en un contexto donde la mano de obra necesaria puede ser limitada y cara) para corregir los errores en vez de las tareas planeadas generando **retrasos**, y puede causar **perdida de clientes y ganancias** (por coste de confianza y/u oportunidad).

5. Clasifique los seis atributos de calidad fundamentales mencionados en el capitulo 1

> Los atributos de calidad de software pueden clasificarse en **externos** (observables por los usuarios del software) e **internos** (concernientes a los desarrolladores).

Externos: Funcionalidad, confiabilidad, usabilidad, eficiencia.

Internos: Mantenibilidad, portabilidad, eficiencia.

6. En el capitulo 1 se explica que una medida comúnmente utilizada para la calidad es defectos por KLOC en el software entregado. Para un producto de software dado, ¿como puede medirse su calidad? ¿como puede ser estimada su calidad antes de ser entregado?

> **LOC:** Lines Of Code

- Por tests pasados, cada uno de ellos verifica que cumple una función pedida. (funcionalidad)
- El uso satisfactorio del producto por parte de 6 usuarios al azar. (usabilidad)
- Probarlo en uso (ej: una beta) en diferentes entornos controlados o semi controlados con recursos variables probando su desempeño. (portabilidad, eficiencia y confiabilidad)
- Pedir una nueva característica no esperada y medir el tiempo y los cambios para su realización. (mantenibilidad)

7. ¿Cual es la diferencia fundamental entre las fallas mecánicas y las de software?

- Que las fallas de software pueden estar desde el inicio. Sin factores externos no surgen con el tiempo.
- Las de software suelen ser mas fáciles de arreglar.

8. Suponga que durante el desarrollo de un proyecto de software se tuviera tiempo extra para mejorar la confiabilidad del producto final. ¿En que utilizaría ese tiempo extra?

> **Confiabilidad:**
>
> - Capacidad de realizar las funciones bajo las condiciones establecidas durante un tiempo especifico.
> - Cuando un programa se comporta de la misma manera a lo largo del tiempo en un mismo sistema operativo.

Testear el producto durante mas tiempo. hacer mas tests.

9. Explique los tipos de calidad que presenta el estándar ISO

- **Calidad en Uso:** Evalúa, desde la perspectiva del usuario final, cómo el software les permite alcanzar sus objetivos en un contexto específico de uso; eficacia, productividad, seguridad, satisfacción.
- **Calidad del Producto:** Evalúa las propiedades internas del software que afectan su calidad y rendimiento; funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad.
- **Calidad en el Proceso:** Se centra en la calidad de los procesos de desarrollo, implementación y mantenimiento del software; cumplimiento, eficiencia, mejora continua.

10. ¿Cuales son los desafíos de la Ingeniería del Software? Explique cada uno de ellos

- **Escala:** Los métodos de IS deben tener la **capacidad de adaptación y respuesta** de un sistema con respecto al rendimiento del mismo a medida que **aumentan o disminuyen** de forma significativa el número de usuarios o requerimientos del mismo. Dimensiones a considerar: Métodos de ingeniería, Administración del proyecto.
- **Productividad:** A mayor productividad, menor costo y/o menor tiempo.
- **Calidad:**
  - Funcionalidad: Capacidad de proveer funciones que cumplen las necesidades establecidas o implicadas.
  - Confiabilidad: Capacidad de realizar las funciones requeridas bajo las condiciones establecidas durante un tiempo específico.
  - Usabilidad: Capacidad de ser comprendido, aprendido y usado.
  - Eficiencia: Capacidad de proveer desempeño apropiado relativo a la cantidad de recursos usados.
  - Mantenibilidad: Capacidad de ser modificado con el propósito de corregir, mejorar, o adaptar.
  - Portabilidad: Capacidad de ser adaptado a distintos entornos sin aplicar otras acciones que las provistas a este propósito en el producto.
- **Cambio:** Las prácticas de IS deben preparar al software para que éste sea fácilmente modificable.
- **Consistencia y repetitividad:** Asegurar que el éxito pueda repetirse, con el fin de mantener alguna consistencia en la calidad y la productividad. Con el objetivo de tener una sucesiva producción de sistemas de alta calidad y con alta productividad. La consistencia permite predecir el resultado del proyecto con certeza razonable. Sin consistencia sería difícil estimar costos.

11. ¿Cuales son las fases del proceso de desarrollo? ¿Cuales la ventaja de dividirlo en fases?

El motivo de separar en fases es la **separación de incumbencias**: cada fase manipula distintos aspectos del desarrollo de software. Ademas, el proceso en fases permite **verificar la calidad y progreso** en momentos definidos del desarrollo (al final de la fase).

En general las fases del proceso de desarrollo consisten de:

1. Análisis de requisitos y especificación
2. Arquitectura
3. Diseño
4. Codificación
5. Testing
6. Entrega e instalación

12. ¿De que manera un proceso separado en fases ayuda en la obtención de alta C&P (calidad y productividad)? ¿Cuando parece que estamos haciendo mas tareas en un proceso por fases que en uno ad-hoc?

> Cada fase termina con una salida definida y se realiza en el orden especificado por un modelo elegido.

Un proceso separado en fases aumenta la productividad debido a que se puede planear mas fácilmente y llevar a cabo de forma mas especializada y ordenada. Un proceso separa en fases aumenta la calidad debido a que ademas de ser mas productivo, también es mas fácil no cometer errores, validar las fases y no olvidar nada.

Parece que estamos haciendo mas cuando planeamos las fases, elegimos el modelo, administramos el proceso,...

13. Entre los atributos de calidad enumerados no se encuentra la reutilizabilidad. Defina este atributo, y describa cual es la relación entre este y portabilidad

> **portabilidad:** Capacidad de ser adaptado a distintos entornos sin aplicar otras acciones que las provistas a este propósito en el producto.

reutilizabilidad: Capacidad de ser reutilizado en distintos entornos con fines diferentes a los iniciales.

Relación: Ambos hablan de la capacidad con la que se puede adaptar de alguna manera a diferentes entornos.

14. ¿Que tipo de mantenimientos necesita un software?

Mantenimiento correctivo: (updates) corrige errores residuales.

Mantenimiento adaptativo: (upgrades) mejora o adapta a cambios en el entorno.