Universidad Nacional del Sur
Departamento de Ciencias e Ingeniería de la Computación

Tecnología de Programación

(2019)

Práctico 2
Ingeniería de Software

Temas: Ingeniería de Software, Ciclo de Vida, Etapas, Riesgos, Factores de Calidad.

Ejercicio 1. Defina "Ingeniería de Software". ¿Qué diferencias encuentra entre la ingeniería de software y otro tipo de ingeniería? ¿Por qué se considera una "Ingeniería"?

Ejercicio 2. ¿Qué es un Ingeniero de Software? ¿Qué perfil debe tener?

Ejercicio 3. Encontrar ejemplos concretos de dependencias de aplicaciones para con el sistema operativo o con el hardware. ¿Por qué razón estas dependencias pueden afectar la calidad del software?

Ejercicio 4. Uno de los sistemas operativos más utilizados en el mundo es Microsoft Windows XP. Evaluar el cumplimiento de los distintos factores externos de calidad de software para este sistema operativo. En este sitio: http://www.engadget.com/2009/08/12/windows-7-review/ se puede leer un "review" de Microsoft Windows 7 ¿Cómo relaciona los puntos a favor y en contra dichos en tal artículo con los factores de calidad externos vistos en clase? ¿Observa diferencias, en cuanto a factores de calidad, con respecto al "review" de Windows 8 http://www.engadget.com/2012/10/30/windows-8-review/? En el mismo sitio se encuentra también un "review" de un sistema operativo de Apple, Snow Leopard (http://www.engadget.com/2009/08/26/snow-leopard-review/). ¿Qué comparación puede hacerse a nivel de los factores de calidad entre los tres sistemas operativos?

Ejercicio 5. ¿Es posible medir la interfaz de un módulo? Proponer al menos dos criterios que sirvan a tal efecto.

Ejercicio 6. Describir un sistema que no sea de software pero que presente una organización basada en módulos, donde toda comunicación entre los distintos módulos se realice a nivel de las interfaces. El sistema descripto, ¿en qué forma cumple las reglas y principios de modularidad?

Ejercicio 7. ¿Es suficiente con que un lenguaje de programación soporte el concepto de módulo, o en realidad hacen falta herramientas meta-lingüísticas (esto es, herramientas fuera del lenguaje) adicionales para asegurar la calidad del software?

Ejercicio 8. Un error común en Ciencias de la Computación es pensar que la codificación insume la mayor parte del tiempo de desarrollo del software. Apelando a la experiencia individual, establecer qué porcentaje de tiempo requieren aproximadamente las fases de especificación de requerimientos, diseño, codificación, testeo y mantenimiento en el desarrollo de un programa.

Ejercicio 9. Definir el concepto de riesgo en el desarrollo de software. Enumerar algunos de los riesgos que deberían tenerse en cuenta a lo largo de la producción de software.

Ejercicio 10. Explicar brevemente cómo funciona el modelo en espiral del ciclo de vida del software. Los modelos en espiral o en cascada (con feedback), ¿se pueden usar indistintamente? Justificar la respuesta suministrada.

Ejercicio 11. Las subrutinas (por ej., los procedimientos y las funciones) son estructuras modulares convencionales. ¿Por qué razón sería incorrecto afirmar que los procedimientos y funciones en Pascal son en general reusables?

Ejercicio 12. La siguiente situación "Si a es verdadero entonces llamar a X y sino llamar a Y", programada en Java quedaría así:

```
if (a) {
1.          X() }
else {
1.          Y()
}
```

¿Cómo se implementaría la misma situación en PASCAL, MODULA-2, C#, C++, EIFFEL y R?

Ejercicio 13. La siguiente situación "Mientas m sea mayor a 10 entonces llamar a X e incrementar m", programada en Java quedaría así:

```
while (m>10) {
1.          X();
1.          m++;
}
```

¿Cómo se implementaría la misma situación en PASCAL, MODULA-2, C#, C++, EIFFEL y R?