PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IIC2113 - Diseño Detallado de Software (2020-2)

Pauta - Interrogación 1

Pregunta 1 [15 puntos totales]

Responda las siguientes preguntas justificando el por qué de su respuesta (3 puntos cada respuesta). Además explique cómo llegó a esta conclusión (2 puntos).

a) ¿Cuáles son los riesgos de tener solo un desarrollador programando un proyecto de desarrollo? Nombre al menos 3 riesgos relacionados con la materia y lecturas del curso. [5 puntos]

i) No se asegura la calidad, ya que los criterios de calidad están definidos solo por el desarrollador (quality design guidelines).
ii) En caso de que el desarrollador enferme o tenga un problema, se verá directamente reflejado en el proyecto, al no tener apoyo (lectura developer isolation).
iii) En caso de que ese desarrollador deje el proyecto, nadie más sabrá cómo abordarlo (problema del experto, clases de buenas prácticas).
iv) Tendencia a encontrar de forma tardía bugs o errores en el código, al no existir un equipo que apoye (clases de buenas prácticas).

El puntaje se reparte: 1 punto por cada razón correcta y explicada. Si no se explica bien, son 0,5. Luego son 2 puntos por justificar en total. Descontar 0,5 por cada razón no justificada.

b) ¿En qué etapa se considera el diseño de software en las metodologías ágiles? Considere al menos 5 líneas para su explicación (aparte de cómo llegó a su conclusión). [5 puntos]

Esta era una pregunta con trampa, porque se considera durante todo el proceso ágil. Si bien, en la etapa inicial se considera en mayor parte y es donde más decisiones se toman, durante la implementación siguiente se sigue diseñando el código y tomando decisiones. Además al ser un proceso iterativo el diseño es algo que va cambiando también.

Basta con mencionar y explicar que se considera durante todo el proceso para tener los 3 puntos. En las 5 líneas se espera que explique cómo funcionan las metodologías ágiles y cómo el diseño de software se involucra (2 puntos).

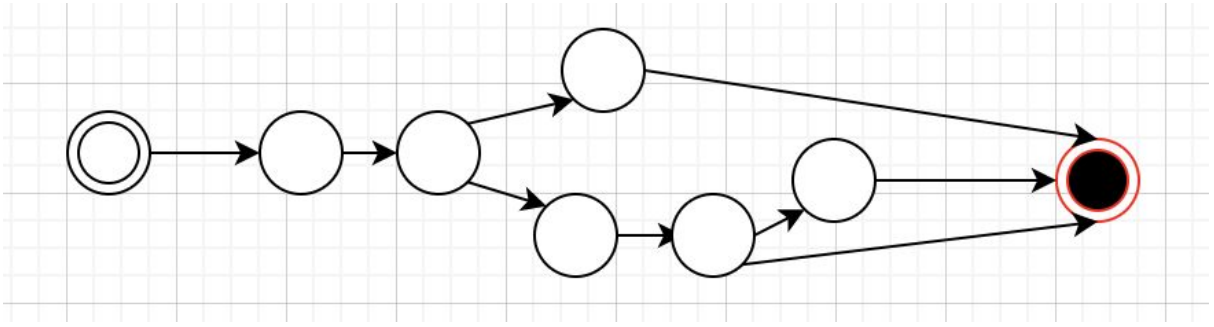
c) ¿Cuál es la complejidad ciclomática de la siguiente función en Ruby? [3 puntos] ¿De qué nos sirve tener esta métrica? [2 puntos]

i) Complejidad 3. Grafo y formula. (1 punto cada uno).

1) Consideren que algunos dejaron como nodo cada declaración. Dado que nodos y aristas se restan y es línear, se llega al mismo resultado. Ambas formas de dibujar el grafo o considerar los nodos y aristas estará correcto, ambos terminan dando la misma complejidad.

ii) Mide la complejidad de un método o programa. A mayor complejidad ciclomática mayor es la cantidad de ramificaciones.

Ejemplo de grafo, otros grafos con más aristas y nodos lineales deberían solucionar también.



$$CYC = E - N + 2 * P$$

$$CYC = 9 - 8 + 2 * 1 = 3$$

Pregunta 2 [10 puntos totales]

a) Nombre y describa al menos 3 problemas que observe en base a los principios fundamentales, TPM, SOLID o Bloaters [6 puntos].

i) De TPM

1) El código invita al desorden, no tiene una estructura lógica clara. Existe un código que no se utiliza, como librerías.

2) Las convenciones de nombre están poco claras, y no se entiende el por qué (variables con nombres de letras).

ii) De SOLID

1) Single responsibility principle, la clase principal tiene todas las responsabilidades: procesa entradas y salidas, parser de datos, y tiene el algoritmo principal.

iii) De bloaters

1) Long method, está todo en el mismo método.

2) Primitive Obsession, uso excesivo de variables de tipo int.

3) Data Clumps, existe mucha replicación del código en la parte del algoritmo y en la lectura de los inputs.

2 puntos cada uno de los 3 problemas (si pone más, elegir los 3 más acertados). Descontar 1 punto por cada razón mal justificada.

b) Proponga una solución a uno de los problemas identificados [4 puntos]. Puede describir detalladamente la solución (apoyándose en algún diagrama) o re-escribiendo la sección de código señalada.

i) Si la solución es a nivel de explicación, el diagrama que acompañe debe tener las explicaciones claras. En general para solucionar SPR o Long method.

ii) Si utiliza código, debe resolver correctamente la sección en todo el código. Por ejemplo, si decide cambiar el nombre de las variables, estas deben ser entendibles y fácilmente diferenciables. Si decide resolver algún bloater, debe ser específico en el código y la solución.

Código de ejemplo aparte.

Pregunta 3 [15 puntos totales]

a) ¿Qué recomendaciones le daría a su amigo en cuanto a las decisiones de diseño que ha tomado inicialmente? [3 puntos]
   i) *No elegir la tecnología antes de decidir cómo abordar el problema o cómo se diseñará. La tecnología debería apoyar al diseño de la solución, el diseño no se debería acoplar a la tecnología a no ser que exista una restricción extra al respecto, y por el enunciado, tu amigo tomó la decisión sin muchos fundamentos reales.*
   ii) *Si enfoca la respuesta al diseño de arquitectura, o al diseño de interfaz, o al diseño de datos dar un punto por cada uno si los explica todos. Considerar -1 puntos si se enfoca solo en uno -> una recomendación de diseño debería ser general y no enfocarse solo en un punto, ya que podría dejar varios elementos o restricciones fuera de su solución.*

b) Ayude a su amigo creando un diagrama de actividad en UML [6 puntos]
   i) *Ojo en las consideraciones que se tomen (si esta la sesión iniciada, si hay un usuario registrado, etc.-)*
   ii) *Estará malo si deciden hacer el diagrama de actividad de un inicio de sesión o de registro (porque no responden al problema de su amigo).*
   iii) *Pueden basarse en el ejercicio práctico.*
   iv) *Se tiene que entender la notación (mitad del puntaje si no se entiende el diagrama).*

c) ¿Qué diagramas recomendaría usar en base a UML, para definir el resto de la aplicación de forma completa? ¿Por qué? [3 puntos]
   i) *Se considera bueno si se recomienda en base al modelo 4+1 otros 3 diagramas.*
   ii) *Se considera bueno si se recomienda en base a la división de UML estructural y de comportamiento.*
   iii) *Se considera bueno si se recomienda en base al diseño en el desarrollo (de las clases iniciales de motivación).*
   iv) *Descontar 1 punto si no explica por qué la elección de esos diagramas (decisiones "al ojo" pierden este punto).*

d) Su amigo no sabe lo que es UML. ¿De qué forma podría explicarle el diseño de la aplicación con el fin de que entienda? [3 puntos]
   i) *Si realiza una explicación detallada de UML y explica sus diagramas con mucho detalle considerar correcto. Sin embargo, el objetivo es pensar un poco más allá de UML. Se espera que le de recomendaciones o explique qué es el diseño de una forma más amigable y en sus propias palabras.a*
      1) *Recomendarle tomar clases de UML o "mandarlo a estudiar" se considera como incorrecto, la idea es explicarle aquí.*
      2) *Explicar el mismo problema que su cliente escribió estará incorrecto.*