Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

28 de junio del 2018

IIC2143 – Ingeniería de Software

Examen

Instrucciones:

- Sea preciso: no es necesario escribir extensamente pero sí ser preciso.
- En caso de ambigüedad, utilice su criterio y explicite los supuestos que considere convenientes.
- Responda y entregue cada respuesta en hojas separadas. Si no responde una pregunta, debe entregar de todas formas la hoja correspondiente a la pregunta.
- Indique su nombre en cada hoja de respuesta.
- Escriba su número de lista en la esquina superior derecha de cada hoja de respuesta.
- Este examen fue diseñado para durar 120 minutos.

1. (0.6 pts)
   a. Comente 3 razones de por qué se utilizan procesos de desarrollo en proyectos de *software*.

      - Sistemas son complejos: diversos actores, servicios y herramientas involucradas
      - Se busca que el valor sea mayor al costo:
        - Planificar y diseñar cuidadosamente
        - Revisión del producto dentro de metas y plazos
      - Replicar éxitos y evitar fracasos:
        - Los procesos son replicables
        - El resultado es más confiable y seguro
      - Visibilizar avances del proyecto

   b. Explique a qué se refiere el concepto de *Minimum Viable Product*. ¿Con qué principio del desarrollo ágil se relaciona?

      El concepto *MVP* se refiere a desarrollar un producto con las funcionalidades justas para satisfacer necesidades claves de los usuarios, además de proveer retroalimentación temprana para los desarrolladores del producto.

      Este concepto se puede relacionar con el desarrollo iterativo de los procesos ágiles, donde se busca que el resultado de cada iteración sea un producto funcional que agregue valor a los usuarios, sin importa que este sea simple o lejano al objetivo final.

2. (0.3 pts) ¿A qué se refiere el "triángulo de triple restricción" en la gestión de proyectos? Explique qué representa cada vértice y sobre qué concepto del proyecto inciden.

   Cada vértice del triangulo representa alcance, tiempo y costos. La ponderación de estas variables incide en la calidad del resultado.

3. (0.6 pts) Explique la práctica conocida como *Test-Driven Development*, detallando sus características principales, junto con 2 beneficios y una desventaja.

   Test-Driven Development es una práctica en el desarrollo de *software* que se basa en la repetición de pequeños ciclos de desarrollo. Al comienzo de un ciclo se deben escribir *tests* de una funcionalidad acotada en especifico. Luego, se debe escribir solamente el código necesario para que dichos *tests* pasen. Una vez que los *tests* están aprobadas, se revisa a través de técnicas de *refactoring* si la misma funcionalidad se podría lograr de mejor manera.

   Beneficios:
   - Fomenta la creación de *tests* y código modular, al abordar las funcionalidades en pequeñas iteraciones.
   - Mejor entendimiento, documentación y diseño de las funcionalidades, ya que requiere definir las pruebas y casos bordes antes de programar.

   Desventajas:
   - Tiene un costo de aprendizaje asociado para implementar la práctica de manera correcta (y que genere realmente valor).
   - Puede ralentizar el desarrollo en proyectos cortos o exploraciones de ideas/conceptos, ya que exige la creación de pruebas.
   - Exige que por lo menos cada desarrollador sea responsable de escribir buenas pruebas y mantenerlas.
   - Difícil de aplicar en código legado sin pruebas.

4. (1.0 pts) Responda si las siguientes afirmaciones son verdaderas o falsas. Justifique las respuestas que considere falsas.

   a. El término "diagrama de dominio" es otra forma de nombrar a un diagrama de "entidad-relación".

      Falso: Son 2 diagramas con objetivos distintos. El primero busca representar las entidades de un problema, mientras que el segundo sirve para modelar datos.

   b. La verificación de un sistema de *software* corresponde a revisar que este haga lo que se especificó que hiciera.

      Verdadero

   c. En la metodología *SCRUM*, la duración de un *sprint* puede estar determinada por la cantidad de *story points* a realizar.

      Falso: La duración de un *sprint* está determinada por el tiempo que se le asigne.

d. Para un equipo ágil en fase de maduración *performing* no es necesario escribir documentación del proceso de desarrollo.

   Falso: Puede ser que se necesite menos documentación, pero siempre será necesaria.

e. El objetivo de las métricas de *software* asociadas a la codificación sirven para proyectar sus costos.

   Falso: Puede que algunas métricas ayuden a proyectar costos, pero el objetivo de todas las métricas es representar el estado del código, para poder comparar la evolución de este.

5. (0.3 pts) Comente 5 características de un equipo de alto rendimiento.

   - Sinérgicos
   - Compromiso con el equipo
   - Comunicación efectiva entre miembros
   - Confianza en las personas y el trabajo
   - Visión y objetivo compartido

6. (0.8 pts) Caso de *Twitter*

   Originalmente el sistema de *software* de *Twitter* se basaba en una gran aplicación en *Ruby on Rails*. Cuando el sistema empezó a tener problemas de disponibilidad, la empresa destinó recursos en mejorar el lenguaje *Ruby*, acelerando su desarrollo y fortaleciendo aún más su comunidad. Por otra parte, incrementaron constantemente la cantidad de máquinas en la infraestructura del sistema, para así responder a la creciente demanda de usuarios. En forma paralela, optimizaron fragmentos del código que eran responsables de los principales cuellos de botella del sistema.

   Tiempo después, dado que los problemas persistían, descompusieron la aplicación en un conjunto de servicios montados sobre servidores de aplicaciones, cada uno corriendo con una base de datos adecuada a sus necesidades.

   a. Comente 2 argumentos que podrían justificar la implementación del sistema original en *Ruby on Rails*.

      - En un principio no había certeza del éxito de la aplicación, por lo que decisiones de escalabilidad o eficiencia en base a las tecnologías no eran relevantes.
      - Un *framework* basado en convenciones (como *RoR*) junto con un lenguaje expresivo (como *ruby*) proveen gran facilidad y agilidad para el desarrollo.

b. ¿Qué puede comentar sobre las 3 medidas iniciales llevadas a cabo por el equipo de ingeniería de la empresa?

- Mejorar el lenguaje: permiten mejoras de *performance* principalmente. También permite fortalecer y fomentar el crecimiento de la comunidad de *ruby*, lo cual facilita la incorporación de nuevos integrantes al equipo de desarrollo y la aparición de mejores prácticas de desarrollo con esta tecnología.
- Incrementar cantidad de máquinas: medida "parche" para evitar el problema ingenieril de fondo.
- Sobre-optimización de código: medida temporal que dificulta la legibilidad y la flexibilidad del código.

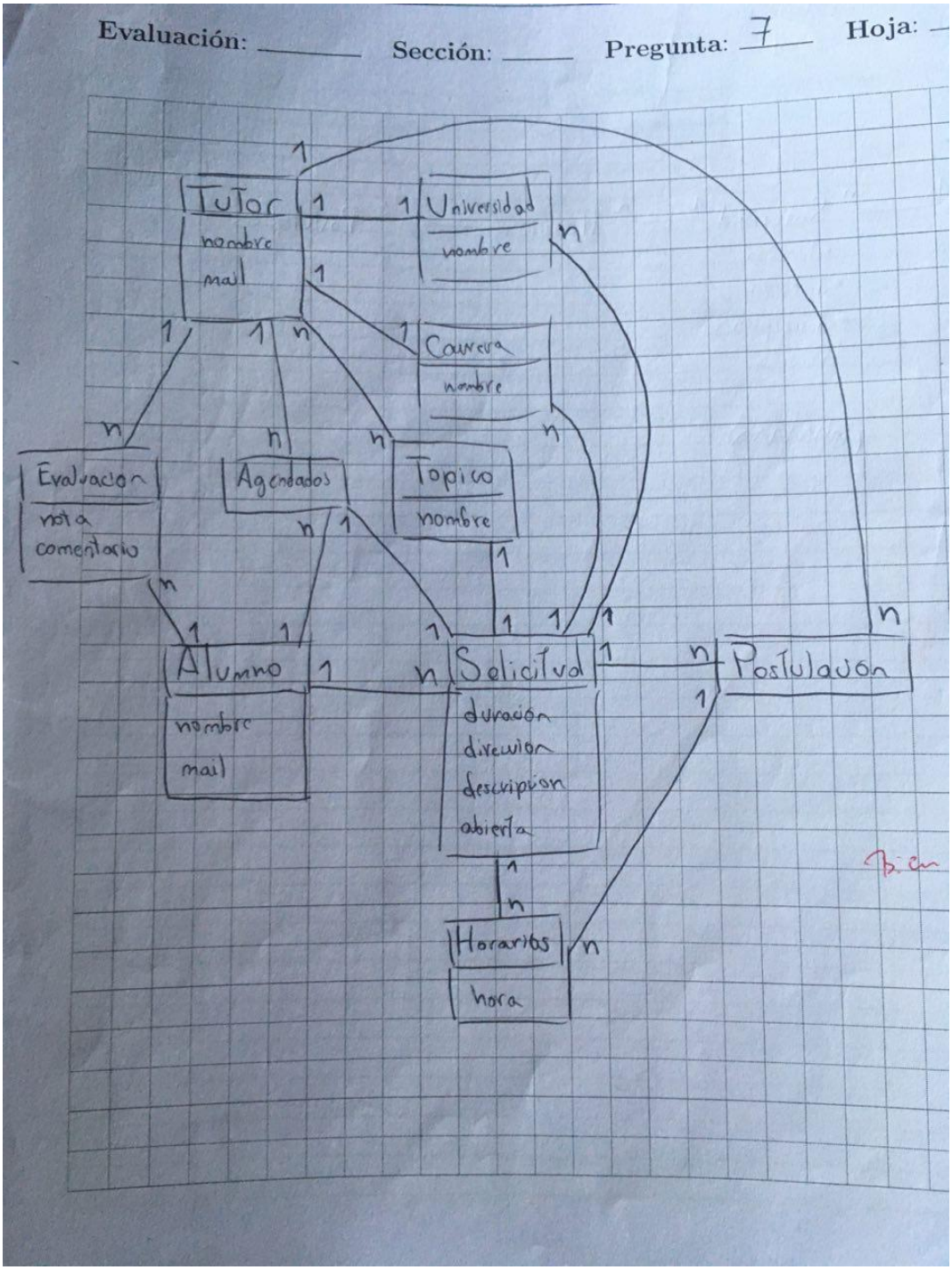
Considere el siguiente contexto para las preguntas 7 y 8:

"Un emprendedor le propone modernizar los sistemas de su empresa, que ofrece coordinación de clases particulares en Chile. Hasta ahora este negocio se gestiona manualmente y la idea es que se pueda garantizar la selección del tutor más adecuado para un alumno de manera automática.

La empresa cuenta con un equipo de tutores pertenecientes a distintas carreras y universidades, los que estén autorizados para realizar clases sobre ciertos tópicos en específico. Cuando un alumno compra una clase se levanta una solicitud para encontrar un tutor. Esta solicitud contiene los posibles horarios de la clase, su duración, la dirección en que se realizará, el tópico a revisar, una breve descripción y restricciones sobre la universidad y/o carrera del tutor que se necesita. Los tutores pueden postular a realizar la clase solicitada, detallando cuáles horarios tienen disponibles. Luego de que existan tres postulantes para una solicitud se revisa si entre estos existe un tutor que esté autorizado para realizar el tópico solicitado y que cumpla con las restricciones de universidad, carrera y horarios de la solicitud. Si estas condiciones se satisfacen se procede a agendar la clase con el tutor seleccionado y se cierra la solicitud. Por último, una vez realizada la clase se le pide al alumno que evalúe con una nota general al tutor, lo que permite mejorar la calidad del servicio ofrecido.

Es importante que se almacene la información histórica de solicitudes, postulaciones, clases realizadas y evaluaciones, para así mejorar los algoritmos recomendadores y de selección del sistema."

7. (1.2 pts) Para dimensionar la complejidad de un *software* que apoye el negocio descrito, se le solicita que realice un **diagrama de dominio** de una posible solución. Su diagrama debe ser *UML 2.0* válido, detallando las entidades (junto con sus atributos relevantes) y los tipos de relaciones entre ellas (junto con su cardinalidad). Su propuesta debe permitir que se realicen todas las funcionalidades que se detallan en el contexto.



8. (1.2 pts) Un problema que ocurre al coordinar manualmente las solicitudes es que si alguno de los requisitos de estas se modifica, es muy costoso notificar a cada postulante de este cambio. Por esta razón se le solicita que cuando se diseñe el *software* se considere que incluya una solución para este requisito.

    a. Escoja un patrón de diseño *GoF* para representar la funcionalidad descrita.
    b. Realice un diagrama *UML 2.0* válido para representar la implementación del patrón. Debe ser explícito y detallado sobre los componentes, métodos y relaciones presentes en el diagrama (considerando el contexto del problema).