CURSO 2016 - 2017

CRITERIOS DE CORRECCIÓN DE LA ACTIVIDAD EVALUABLE Y CALIFICABLE

1. Consideraciones generales.

En este tipo de ejercicios no hay una solución única y cerrada, las respuestas deben considerarse en el contexto de **lo útiles que sean**. La elaboración del análisis y diseño debería recoger la mayoría de los aspectos '*principales y obvios*' e intentar transmitir, con claridad, qué es lo importante que se debe saber acerca del problema y sus soluciones. Es necesaria cierta experiencia y criterio formado para evaluar en qué grado se han alcanzado estos objetivos.

Hay que tener en cuenta que los Casos de Uso no tienen ninguna relación con la orientación a objetos por lo que, de su elaboración, no se desprende ninguna medida de las capacidades del estudiante relacionadas con el objetivo central de esta asignatura. Por el contrario, las habilidades mostradas en la asignación de responsabilidades y en el diseño de las colaboraciones, son las de mayor importancia porque:

- Son ineludibles.
- Influyen fuertemente en la calidad, escalabilidad y mantenibilidad de los sistemas software.

Juan del Rosal, 16
28040, Madrid

Tel: 91 398 89 10
Fax: 91 398 89 09

swdesign@issi.uned.es

2. Respuestas.

Evaluación de ***Casos de Uso***

1. (0'5 puntos) En relación a las operaciones en la aplicación e-Vuela, identifique al menos 4 casos de uso primarios y sus actores correspondientes. Represente los resultados en un diagrama de casos de uso de UML.

   Se debería obtener algo similar a los servicios que provee la aplicación, a los usuarios genéricos o a alguno específico, y que se indican en el enunciado. Un aspecto muy importante es saber diferenciar los casos de uso primarios (operación o servicio principal del subsistema software que se está analizando, normalmente se corresponde con una ***funcionalidad esencial*** del sistema) de las operaciones, funciones y casos de uso secundarios. Por ejemplo, el *acceso* y la *identificación* respecto a la '*consulta de itinerarios*' o a la '*compra de un viaje*'.

   Otro aspecto importante es saber delimitar qué funcionalidades están *dentro* del sistema que se analiza (es decir, que parte del negocio del cliente es la que se va a *traducir* a software) y con qué elementos interactúa (actores); ya sean humanos, físicos automáticos u otro software.

   De otra forma, posiblemente el estudiante no tenga las habilidades suficientes para identificar procesos primarios. También es posible que no haya manejado nunca la reserva de un vuelo o la compra de un viaje por Internet. Este enunciado ilustra adecuadamente la situación en la que, el analista, desconoce los detalles del negocio del cliente. Ese conocimiento es responsabilidad del ingeniero y es fundamental para la comunicación con el cliente y la obtención de un producto aceptable.

   Los casos de uso suelen nombrarse empezando con un verbo.

Errores

Un error común es representar un *paso* o una secuencia de un caso de uso como un caso de uso separado. Por ejemplo, no son aceptables los casos de uso *Imprimir billetes*, *Registrar datos de pago* o *Autentificarse*. Por supuesto, los casos de uso abstractos que se manejan en las relaciones de <<*uso*>> puede que sean procesos o pasos individuales, pero no es aplicable en este contexto, sino en la factorización de

subprocesos comunes que provienen de casos de uso primarios múltiples.

2. (1 punto) Escriba el caso de uso <<*ProcesarCompra*>> en un formato completo (se recomienda la variante '*en dos columnas*') y estilo esencial. Incluya tanto el escenario principal de éxito como 2 extensiones o flujos alternativos que pudieran ser frecuentes. Suponga que el escenario de partida para este caso de uso es el de un usuario registrado que ya está identificado en el sistema, que ya había realizado varias consultas para un viaje, para un mínimo de 2 viajeros, y las había almacenado en su perfil. Dicho usuario decide realizar la compra correspondiente a un itinerario, que ya había reservado, de una de las listas de itinerarios que ha recuperado de su perfil (el flujo básico de acciones comienza con la orden de compra del itinerario reservado y termina con la obtención de los billetes). En este flujo principal, considere sólo el pago por medios electrónicos, sin tener en cuenta las políticas de precios, descuentos o promociones de la agencia de viajes. No escriba un encabezamiento demasiado elaborado del caso de uso (es decir, omita *propósito*, *resumen*…); en su lugar, afronte directamente el transcurso típico de los acontecimientos.

Caso de uso: **Procesar Compra**

Estilo informal, extendido y esencial.

Evolución típica de los acontecimientos

- Debería tener un formato en dos columnas: la de las *Acciones del Actor* y la de las *Respuestas del Sistema*.
- El primer paso debería comenzar con inicio canónico "El caso de uso comienza cuando el <*Actor*>…".
- No debería incluir alternativas, sino solamente la típica sucesión de acontecimientos de la compra satisfactoria de un viaje, sin problemas. Es un aspecto muy importante para la calificación, puesto que es la idea central de los casos de uso.
- Puesto que se especifica 'estilo esencial', se deben evitar detalles relacionados con los diversos medios y formatos de entrada/salida (como lectores de tarjetas, apretar botón, etc.). La distinción entre casos de uso esenciales y reales se debe reflejar en la expresión del estudiante mediante cierto nivel de abstracción.

Ejemplos de Alternativas muy frecuentes

- Las situaciones o flujos alternativos que son muy frecuentes (respecto al transcurso satisfactorio del caso de uso), deben encontrarse entre las alternativas. Su ausencia puede reflejar un análisis muy pobre.

Evaluación del ***Modelado Conceptual***

3. (2 puntos) En relación al caso de uso anterior *<<ProcesarCompra>>*, construya un Modelo de Dominio y represéntelo en notación UML. Represente los objetos conceptuales, las asociaciones y los atributos.

   - Ya que es una herramienta de comunicación, hay una enorme variabilidad aceptable en el modelo conceptual; que es aún más marcada en las asociaciones.
   - La diferencia entre las multiplicidades "*" y "1.." es controvertida y puede ser muy discutible en un examen. Sin embargo la distinción suele ser relevante en un proyecto real en tanto en cuanto las comprobaciones de la coherencia en las referencias se realizan mediante las multiplicidades válidas.
   - Como se pide el Modelo de Dominio restringido al caso de uso *ProcesarCompra*, no se debería extender en conceptos mucho más allá de este caso de uso particular. Por ejemplo, aunque en la compra se está utilizando un *itinerario*, es mucho menos relevante cómo se ha obtenido que la información que está manejando y, sobre todo, es más importante que se representen claramente los mecanismos para obtener la información específica que se necesita para realizar una acción (trinomios Catálogo-Descripción-Artículo), o los necesarios para arbitrar y dirigir el tráfico de eventos y mensajes (lo que luego serán *Controladores*) o los requeridos para el almacenamiento y seguimiento de la información trascendental del caso de uso (*Registros*, etc.). La presencia de elementos irrelevantes podría indicar una laguna en la comprensión de cómo limitar el modelo mediante unos pocos casos de uso y, en general, de cómo aplicar el proceso incremental iterativo para limitar el análisis. La ausencia de elementos importantes, como los anteriores, podría indicar un defecto en la comprensión del propio problema o de los sencillos y comunes mecanismos, que se ven en la asignatura, para dar solución a operaciones simples.

Objetos que debe contener

- El modelo debería contener la mayoría de los elementos que aparecen en la solución estándar; especialmente los que articulan la secuencia de pasos de la compra y el registro de la operación.
- Conceptos de menos importancia: los potencialmente relacionados con las consultas y la obtención de la información necesaria para realizar la secuencia de operaciones anterior.
- Al igual que en el dominio de Punto de Venta (PDV), es necesario hacer la distinción de la descripción de un elemento que se maneje en la transacción (es decir, la información que se quiere utilizar del elemento, como el 'número de vuelo' o el 'precio del viaje'), de manera separada al objeto conceptual *en sí* (en el ejemplo, como Vuelo o Itinerario). El estudiante debe reflejar esa distinción.

Errores

Un error indiscutible es usar el símbolo UML de agregación cuando no es apropiado.

Otro, muy frecuente, es anidar la inclusión de objetos para poder acceder a toda la información que se pueda necesitar.

Evaluación de los ***Eventos del Caso de Uso*** (inicio de ***Asignación de Responsabilidades***)

4. (1'5 puntos) Circunscrito al caso de uso anterior <<*ProcesarCompra*>>, construya un Diagrama de Secuencia (diagrama de interacción DS) en UML. Represente los actores y los eventos de los componentes del sistema para este caso de uso.

   A partir de este DS, especifique los contratos de **dos** de las siguientes operaciones: 'IniciarCompra' (inicio de la compra del itinerario reservado), '*AgregarViajero*' (agregar la información de cada viajero para poder emitir los billetes), 'RealizarPago' (pago con tarjeta de todo el viaje, con todos los billetes) o '*ImprimirBilletes*' (emisión e impresión de todos los billetes, de todos los vuelos y de todos los viajeros, tras comprobar el pago).

El DS

- Debería ser un diagrama muy simple y directo; pero se hace hincapié en que el receptor de los eventos no sea el *Sistema* global, sino que se inicie la toma de decisiones para mostrar

el objeto que asume el rol de controlador principal y se diversifique el paso de mensajes entre los objetos principales del Modelo de Dominio.

- Puede haber una gran variabilidad en los nombres de los eventos, pero deben indicar con claridad las operaciones y su naturaleza.
- Los parámetros deben ser como se indica en la solución dada (objetos o atributos definidos en el modelo).

Errores

- El nombre de un evento que no empieza con un verbo.
- Incluir un evento que se produce en una entrada de nivel medio bajo (por ejemplo *introducirDNIViajero(dniViajero)*).
- Utilizar eventos no coherentes con el modelo de dominio definido anteriormente.
- Incluir eventos absurdos o sin sentido.

Los contratos de las operaciones

- Aunque no tienen porqué ser completos o exhaustivos, de ninguna manera pueden ser ambiguos. Según el juicio del estudiante, se pueden añadir postcondiciones o responsabilidades. Pero todas deben estar explicadas y justificadas.
- Durante la fase de diseño, y en los análisis siguientes, no se está limitado a las definiciones originales del modelo de dominio, que no es más que un punto de arranque. Durante la elaboración de los contratos, o de los diagramas de colaboración, se deben añadir al modelo de dominio nuevos objetos, atributos o asociaciones, según se vaya descubriendo la necesidad de su uso. De igual forma, algunos elementos originales del modelo de dominio pueden dejar de ser relevantes en el trabajo posterior.

Errores

- El nombre de un evento que no empieza con un verbo.
- Incluir un evento que se produce en una entrada de nivel medio bajo (por ejemplo *introducirDNIViajero(dniViajero)*).
- Utilizar eventos no coherentes con el modelo de dominio definido anteriormente.
- No expresar las postcondiciones (o responsabilidades) en términos de creación de instancias, creación de asociaciones, asignación de valores, etc. Igualmente, no

expresar las precondiciones en esos mismos términos o que las postcondiciones dependan de precondiciones que no están enunciadas explícitamente.

- Incluir eventos absurdos o sin sentido.

Evaluación de la ***Asignación de Responsabilidades*** y ***Diseño de Colaboraciones***

5. (2 puntos) A partir del contrato de la operación <<*se omite la operación A*>> que haya indicado en el punto 4, complete el diagrama de colaboración en UML. Consigne cada mensaje con los patrones GRASP (Experto, Creador, etc.) o cualquier otro que lo justifique. Si añade responsabilidades no explicitadas en el contrato (porque crea que es importante señalarlas), explíquelas brevemente.

Operación A

- Manejar muchas opciones es algo inherente al diseño y, por ello, debería ser aceptable que haya cierta variabilidad en las soluciones. Lo importante es que el estudiante haga un uso juicioso y sensato de los principios GRASP.
- Debe dejar bien claro (en su denominación y en las responsabilidades que tiene) cuál es el controlador de la operación.
- Una vez definido el controlador, se deben reflejar con claridad (mediante el paso de mensajes) las colaboraciones con otros objetos, las transacciones, las creaciones y las consultas; de manera que se efectúe la operación tal como se indica en el contrato y de manera coherente al transcurso de acontecimientos de la escritura del Caso de Uso.
- Un estudiante experimentado usaría algún patrón (GoF) para manejar las operaciones del sistema; aunque quizás sea una pretensión ambiciosa para uno principiante. (El uso adecuado de patrones GoF aumenta notablemente la calificación, siempre y cuando –claro está— el desarrollo general y la aplicación de los principios GRASP sean correctos).
- Los mensajes a un objeto múltiple (colecciones de objetos) deberían dirigirse al propio objeto y no de forma indeterminada a los miembros de la colección.

Errores

- Los mensajes son invocaciones a responsabilidades que residen en el objeto de destino. Son errores muy comunes:
  a) La clase destinataria del mensaje no tiene asignada esa responsabilidad.
  b) El mensaje no contiene la información necesaria (argumentos) para realizar la acción que invoca. Debido a que la clase receptora no maneja la información que necesita o, ésta, no se recibe en el mensaje, el objeto no puede llevar a cabo esa responsabilidad.
  c) Un objeto necesita una información pero o no se la pide a otro, Experto de esa información, o no aparece como resultado del mensaje (invocación) al Experto.
  d) Una clase acapara gran cantidad de información, por lo que detenta una sobrecarga de responsabilidades. Transgrede varios principios GRASP importantes. Esto también se puede apreciar en el Modelo de Dominio, con una gran cantidad de relaciones de composición y un número elevado de objetos que son 'componentes' de otro (el '*sobrecargado*').
- La secuencia de los mensajes no es coherente con lo descrito en el contrato o con la secuencia de operaciones del Caso de Uso.

6. (2 puntos) A partir del contrato de la operación <<*se omite la operación B*>> que haya indicado en el punto 4, complete el diagrama de colaboración en UML. Consigne cada mensaje con los patrones GRASP (Experto, Creador, etc.) o cualquier otro que lo justifique. Si añade responsabilidades no explicitadas en el contrato (porque crea que es importante señalarlas), explíquelas brevemente.

Operación B

Similar a 5.

Evaluación de los ***Diagramas de Clases*** de diseño

7. (0'5 puntos) Elabore un diagrama de clases para el caso de uso que se está tratando <<*ProcesarCompra*>> (DCD), centrado en la clase cuya responsabilidad es el registro de la compra de la reserva seleccionada y de la emisión de billetes, según se ha descrito en el caso de uso. Represente los nombres de todos sus atributos, asociaciones (con la navegabilidad) y métodos.

Si el diagrama se limita a las colaboraciones de las clases indicadas, debería ser muy simple y directo.

Errores

Un error muy común es que alguna clase contenga una gran cantidad de instancias, de otras clases, como atributos; en lugar de establecer asociaciones con su navegabilidad.

Evaluación de la ***Transformación del Diseño en Código***

8. (0'5 puntos) A partir de los anteriores diagramas de clases y colaboraciones, elabore y defina la clase que haya definido, en el desarrollo anterior, como responsable de la compra de la reserva seleccionada y de la emisión de billetes en el caso de uso <<*ProcesarCompra*>>. Incluya las definiciones de todas las variables que la componen (miembros), pero escriba solamente la definición completa del cuerpo para el método (o métodos) principal/es o más significativo/s: <<*se omite el método*>>. Ignore los pequeños detalles de sintaxis -el objetivo es evaluar la capacidad fundamental para transformar el diseño en código-. Utilice la sintaxis de Java.

   A partir de la definición limitada para clase central, la solución debería ser muy simple. Las variables de las instancias deberían ser privadas. Lo importante es la coherencia con el diseño elaborado.