Universidad de Mariano Gálvez
Facultad de Ingeniería
Ingeniería en Sistemas de Información
Centro de Antigua Guatemala
Curso: Análisis de Sistemas
Catedrático: Ing. Josué Antonio Barillas García
Sección: A

Tarea: Hoja de trabajo 1

Carné: 1290-18-10184 **Nombre:** Angel Fernando Cumez Chipix

Hoja de Trabajo 1

1. Clasificación de los diagramas según la especificación UML 2. Incluya un diagrama diseñado por usted mismo y describa ampliamente las principales dos categorías dentro de esta clasificación.

Existen dos clasificaciones de diagramas: **Los diagramas estructurales y los diagramas de comportamiento**. Todos los diagramas UML están contenidos en esta clasificación.

Diagramas estructurales: Los diagramas estructurales muestran la estructura estática del sistema y sus partes en diferentes niveles de abstracción. Existen un total de siete tipos de diagramas de estructura.

Diagrama de clases: Muestra la estructura del sistema, subsistema o componente utilizando clases con sus características, restricciones y relaciones: asociaciones, generalizaciones, dependencias, etc.

Diagrama de componentes: Muestra componentes y dependencias entre ellos. Este tipo de diagramas se utiliza para el desarrollo basado en componentes (CDB), para describir sistemas con arquitectura orientada a servicios (SOA).

Diagrama de despliegue: Muestra la arquitectura del sistema como despliegue (distribución) de artefactos de software.

Diagrama de objetos: Un gráfico de instancias, incluyendo objetos y valores de datos. Un diagrama de objeto estático es una instancia de un diagrama de clase; muestra una instantánea del estado detallado de un sistema en un punto en el tiempo.

Diagrama de paquetes: Muestra los paquetes y las relaciones entre los paquetes.

Diagrama de perfiles: Diagrama UML auxiliar que permite definir estereotipos personalizados, valores etiquetados y restricciones como un mecanismo de extensión ligero al estándar UML. Los perfiles permiten adaptar el metamodelo UML para diferentes plataformas o dominios.

Diagrama de estructura compuesta: Muestra la estructura interna (incluidas las partes y los conectores) de un clasificador estructurado.

Diagramas de Comportamiento: A diferencia de los diagramas estructurales, muestran como se comporta un sistema de información de forma dinámica. Es decir, describe los cambios que sufre un sistema a través del tiempo cuando está en ejecución. Hay un total de siete diagramas de comportamiento, clasificados de la siguiente forma:

Diagrama de actividades: Muestra la secuencia y las condiciones para coordinar los comportamientos de nivel inferior, en lugar de los clasificadores que poseen esos comportamientos. Estos son comúnmente llamados modelos de flujo de control y flujo de objetos.

Diagrama de casos de uso: Describe un conjunto de acciones (casos de uso) que algunos sistemas o sistemas (sujetos) deben o pueden realizar en colaboración con uno o más usuarios externos del sistema (actores) para proporcionar algunos resultados observables y valiosos a los actores u otros interesados del sistema(s).

Diagrama de máquina de estados: Se utiliza para modelar el comportamiento discreto a través de transiciones de estados finitos. Además de expresar el comportamiento de una parte del sistema, las máquinas de estado también se pueden usar para expresar el protocolo de uso de parte de un sistema.

2. Indique las diferencias entre la especificación UML 2 y la UML 2.5

   los cambios en el diagrama de caso de uso entre las versiones 2.0 y 2.5 son totalmente irrelevantes. El cambio de versión en sí trae algunos cambios editoriales, pero la lógica de los diagramas de CU (y casi todos los demás) es bastante similar. Entonces, a menos que esté realizando una prueba muy avanzada / detallada. Esto quiere decir que son casi idénticos, pero si existen algunas diferencias como por ejemplo (nuevos tipos de diagrama o elementos como puntos en el diagrama de clase)

3. Enumere y describa cada uno de los símbolos utilizados para construir un diagrama de casos de uso. Incluya aquellos que no fueron vistos en clase.
   1. Generalización: Un caso de uso también se puede especializar en uno o más casos de uso hijos.
   2. Caso de uso: se muestra como una elipse que suele incluir un texto describiendo brevemente el proceso.
   3. Sistema: el sistema al que se refiere el caso de uso tiene forma de rectángulo.0
   4. Actor: tanto si es una persona, como un sistema, se representa con el dibujo de una figura humana esquemática.
   5. Asociación <>: el caso de uso en el cual comienza la línea discontinua se relaciona con un segundo caso de uso señalado por la punta de la flecha.
   6. Asociación <>: el caso de uso en el cual comienza la línea discontinua puede extenderse al caso de uso señalado por la punta de la flecha bajo ciertas condiciones, que no han de cumplirse necesariamente en todos los casos.
   7. Inclusión: Un caso de uso puede incorporar el comportamiento de otros casos de uso como fragmentos de su propio comportamiento.
   8. Extensión: Un caso de uso también se puede definir como una extensión incremental de un caso de uso base.

4. ¿Qué criterios utilizaría usted para poder identificar en qué situación utilizar una relación de tipo <extend> y una relación de tipo <include>? No cite o haga referencia a ejemplos para responder esta pregunta. Enumere los criterios.

   1.Extend:
   En el caso de **Extend** la podemos usar cuando el segundo caso de uso no es indispensable que ocurra, y si ocurre le agrega mas valor al caso de uso base u original.
   2.Include:
   En mi forma de pensar la relación **Include**, la usaríamos cuando unimos dos casos de uso que el primero incluye al segundo es decir que el primero no podría funcionar de manera correcta sin el segundo porque no podría cumplir con su objetivo esto quiere decir que el segundo caso de uso es esencial del primer caso de uso.

5. Dado el siguiente problema identifique una cantidad moderada de tareas o actividades que forman parte de dicho proceso. No haga referencia a actores ni casos de uso, mucho menos a cualquier tipo de software. Suponga que en esta etapa usted aun no piensa en construir un software sino que solamente esta interesado en conocer el negocio y su funcionamiento "actual".
   Compra de boleto para viajar en bus extraurbano desde o hacia las ciudades de Guatemala, Antigua Guatemala y Chimaltenango. Considere horarios, precios, horas pico.7
   1. Datos del cliente
   2. Forma de pago
   3. Estación de bus
   4. Horarios
   5. Horas pico
   6. Destinos
   7. Precio de boleto
   8. Tamaño del bus
   9. Numero de asiento

6. ¿Qué es un stakeholder y qué importancia tiene este concepto en la Ingeniería en Sistemas?

   Un Stakeholder es el público de interés de una empresa que permite su pleno funcionamiento. Por público se refiere a cualquier persona u organización que esté relacionada con las actividades y decisiones de una empresa, Por ejemplo, empleados, proveedores, clientes, gobierno. En ingeniería de sistemas se refiere a esta persona u organización que está interesada en llevar a cabo un proyecto o tarea y promoverlo, ya sea a través de su poder de decisión o de financiación, o mediante su propio esfuerzo.

7. ¿Cuáles son las convenciones de nombres más utilizadas y en qué lenguaje de programación son utilizadas cada una de ellas?

   Nombres descriptivos
   Índices y contadores
   Constantes, clases y variables

Estilo de nombrado
Al margen del nombre que utilicemos para nombrar una variable, función u otro elemento, tenemos el estilo o convención que utilizaremos para escribir nombres compuestos por varias palabras.

Existen varias convenciones a utilizar:

Nombre		
camelCase	Primera palabra, minúsculas. El resto en minúsculas, salvo la primera letra. La más utilizada en Javascript.	precioProducto
PascalCase	Idem a la anterior, pero todas las palabras empiezan con la primera letra mayúscula. Se utiliza en las Clases.	PrecioProducto
snake_case	Las palabras se separan con un guión bajo y se escriben siempre en minúsculas.	precio_producto
kebab-case	Las palabras se separan con un guión normal y se escriben siempre en minúsculas.	precio-producto
dot.case	Las palabras van en minúsculas separadas por puntos. En Javascript no se puede usar.	precio.producto

los lenguajes de programación en los cuales se utilizan estos ejemplos de convenciones son:
ActionScript
Ada
C y C++
Java
JavaScript
Lisp
.NET
Objective-C
Perl
Python y Ruby

8. Indique ejemplos de nombramiento de identificadores entre los lenguajes Java, C# y C++. (esto incluye variables globales, variables locales, clases, métodos, funciones, constantes, etc.)

C

auto, const, double, float, Int, short, struct, unsigned, break, continue, else, for, long, signed, switch, void, case, default, enum, goto, register, sizeof, typedef, volatile, char, do, extern, if, return, static, union, while.

JAVA

Abstract, assert , Boolean, break, byte, case, catch, char, class, const, continue, default, double, else, enum, extends, false, final, finally, float, for, goto, if, implements, import, tinstance, of, int, interface, long, native, new, null, package, private, protected, public, return, short, static, strictfp, String, super, switch, synchronized, this, throw, throws, transient, true, try, void, volatile, while.

9. Construya un cuadro comparativo incluyendo las diferencias y similitudes entre una clase concreta, una clase abstracta y una interface.

Clase Abstracta	Interfaz	Clase concreta
La palabra clave abstract se usa para crear una clase abstracta y se puede usar con métodos.	La palabra clave de interface se usa para crear una interfaz, pero no se puede usar con métodos.	Debe proporcionar su propia implementación de métodos abstractos
Una clase puede extender solo una clase abstracta.	Una clase puede implementar más de una interfaz.	Todos los métodos están completamente implementados.
Las variables no son definitivas por defecto. Puede contener variables no finales.	Las variables son finales por defecto en una interfaz.	No hay funciones puramente virtuales.
Una clase abstracta puede proporcionar la implementación de una interfaz.	Una interfaz no puede proporcionar la implementación de una clase abstracta.	Puede ser instanciado
Puede tener métodos con implementaciones.	Proporciona una abstracción absoluta y no puede tener implementaciones de métodos.	Pueden usar clases para instanciar un objeto
Puede tener modificadores de acceso públicos, privados, estáticos y protegidos.	Los métodos son implícitamente públicos y abstractos en la interfaz de Java.	Siempre heredan por lo menos un método abstracto de su superclase
No admite herencias múltiples.	Es compatible con herencias múltiples.	Puede tener subclases abstractas
Es ideal para la reutilización del código y la perspectiva de la evolución.	Es ideal para la declaración de tipo.	Pueden subclases abstractas terminales en el árbol de herencia