INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Modelo			
	Junio 2010			Ámbito
				NACIONAL 1ª SEMANA

ESTE EJERCICIO NO ES DE TEST.

NECESITO EL EJERCICIO MANUSCRITO POR EL ALUMNO Y LA HOJA DE LECTURA ÓPTICA PARA PODER CALIFICAR.

SI ESTE EJERCICIO NO SE HA IMPRESO EN HOJA DE LECTURA ÓPTICA, SOLICITE UNA AL TRIBUNAL.

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En cada pregunta teórica, que se valora con 2'5 puntos, la nota mínima es 1 punto; en la segunda parte (ejercicio de teoría aplicada que se valora con 5 puntos) la nota mínima que debe obtener es de 2 puntos.

¡ATENCIÓN! PONGA SUS DATOS EN LA HOJA DE LECTURA ÓPTICA QUE DEBERÁ ENTREGAR JUNTO CON EL RESTO DE LAS RESPUESTAS.

Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm)

PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. ¿En qué consiste la actividad de reingeniería y cuándo es necesaria?

   Solución

   Consiste en generar un sistema bien organizado y documentado a partir de un producto software que no fue desarrollado siguiendo técnicas de ingeniería de software.

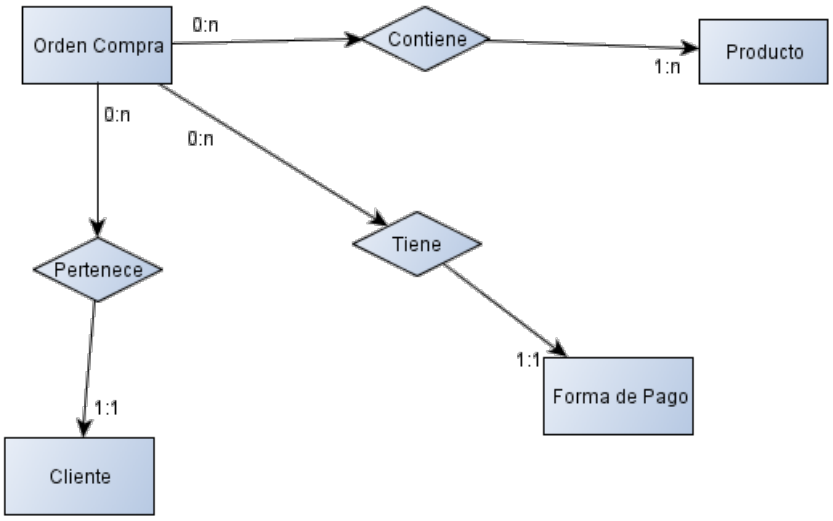
   En ocasiones es necesaria para el mantenimiento de dichos productos software. (Pág. 26)

2. El sistema de ventas por Internet de una tienda funciona de la siguiente manera: para que el cliente formalice la compra debe estar previamente registrado. El formulario de compra consiste básicamente en tres partes: datos del cliente, forma de pago y la lista de los productos seleccionados. Cuando se formalice la compra el sistema guarda dicha operación con: un identificador (orden de compra), el cliente y la lista de productos.

   Realice un diagrama de modelos de datos Entidad - Relación de la compra. Describa los datos más relevantes mediante el ***diccionario de datos***.

Solución

El diagrama E – R para el modelo de datos de la ‘compra’:



Y el diccionario de datos para los elementos más relevantes:

Nombre**:** **Orden de compra**
Estructura**:** **Identificador + Cliente + Producto + {Producto}**
Identificador = {CaracterAlafanumérico}^{N}

Nombre: **Producto**
Estructura: **Nombre + Identificador + Precio**

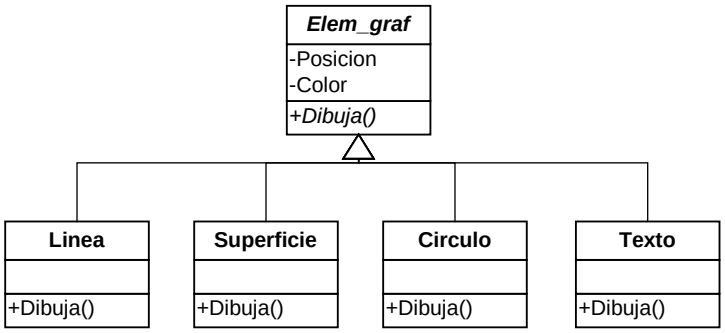
Nombre: **Forma de pago**
Estructura: **[Contrareembolso | Tarjeta | Transferencia]**

Nombre: **Cliente**
Estructura: **Nombre + Apellidos + IdDNI + Usuario + Clave + Dirección**

Nombre = {CaracterAlafnumérico}^{10} **/ristra de 10 caracteres /**
Apellidos = {CaracterAlafanumérico}^{30} **/ristra de 30 caracteres/**
IdDNI = {Dígito}^{8}
Usuario = {CaracterAlafnumérico}^{10}
Clave = {CaracterAlafnumérico}^{10}
Dirección = {CaracterAlafanumérico}^{50}

SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Un módulo o una parte de una aplicación gráfica dispone de elementos gráficos primitivos como `linea`s, cadenas de `texto`, `circulo`s, `superficie`s, etc. Para facilitar el uso de estos elementos, sería deseable que cualquier otro módulo o aplicación cliente pudiera manejarlos de forma similar: los pinte en una posición, los mueva, cambie su color, etc. Para conseguirlo, si disponemos del polimorfismo de herencia, se puede crear una clase abstracta o una interfaz, `Elem_graf`, que recoge las características comunes de todos ellos. Los métodos de `Elem_graf` no tienen contenido sino que, mediante un **enlace dinámico**, se resuelve el código que corresponda al hijo cuando, en tiempo de ejecución, se cree una instancia del objeto utilizado. De esta manera la aplicación cliente no necesita saber la estructura del árbol y le es indiferente tratar con `linea` o con `texto`.



Un escalón adicional en el diseño es comprender que la aplicación cliente querrá manejar `dibujo`s (agrupaciones o compuestos de los elementos gráficos primitivos) de la misma manera que maneja los elementos primitivos.

Para hacer esto hay varias aproximaciones:

1. La más simple. Constrúyase una nueva clase `Contenedor` formada por una colección de los elementos gráficos.

   Pregunta a) A partir del diagrama anterior, represente esta solución con la clase `Contenedor`.

   Nótese que ahora el uso que la aplicación cliente hace de los objetos es análoga a la inicial cuando, al no haber agrupado aún los elementos gráficos primitivos en una familia, no existía el enlace dinámico y tenía que discernir si trataba con `linea` o `circulo`. Ahora `Contenedor` no pertenece a la misma clase que `Elem_graf` por lo que habrá que reescribir el código del cliente de manera que, en lugar de invocar ciegamente el `Dibuja()` de los elementos gráficos hijos, haga:

   a. Compruebe si está tratando con un `Contenedor` o un `Elem_graf`.
   b. Si es un `Contenedor`, obtenga la lista de los hijos contenidos en él e invoque `Dibuja()` de cada uno.

   Los inconvenientes de esta solución son que cada cliente debe preocuparse de cómo está hecha la clase `Contenedor` e incluir código extra para manejar el hecho de que podría estar utilizándola o podría estar usando `Elem_graf`. Además, un `Contenedor` no puede contener `Contenedor`es por el mero hecho de ser una agregación de `Elem_graf`.

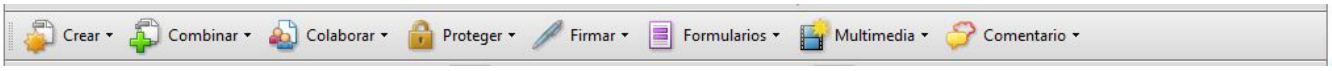
2. Esta segunda solución, que llamaremos **Patrón de Composición**, recoge las ventajas de la solución anterior y resuelve sus inconvenientes. Por un lado, la anterior clase `Contenedor` se va a denominar, definitivamente, `Dibujo`; y es un `Elem_graf` más. Por otro lado, `Dibujo` está compuesto por cualquier `Elem_graf`, incluido él mismo. Las ventajas de esta solución son:

   - No hay que hacer nada especial para que la clase compuesta (`Dibujo`) se contenga a sí misma o a otros elementos de la familia y, así, podemos tener estructuras de cualquier profundidad.
   - Se preserva la simplicidad de los clientes porque sólo deben conocer una clase y no están obligados a replicar en ellos la estructura del árbol.
   - Se pueden añadir nuevas subclases de `Elem_graf`s y también otros tipos de `Dibujo`s, porque los clientes no tienen por qué verse alterados.

Pregunta b) **A partir del diagrama inicial, represente la solución del Patrón de Composición con la clase `Dibujo`.**

Este hallazgo de una solución que resuelve no sólo un problema específico de diseño, sino una buena cantidad de otros problemas similares, se denomina **patrón de diseño**.

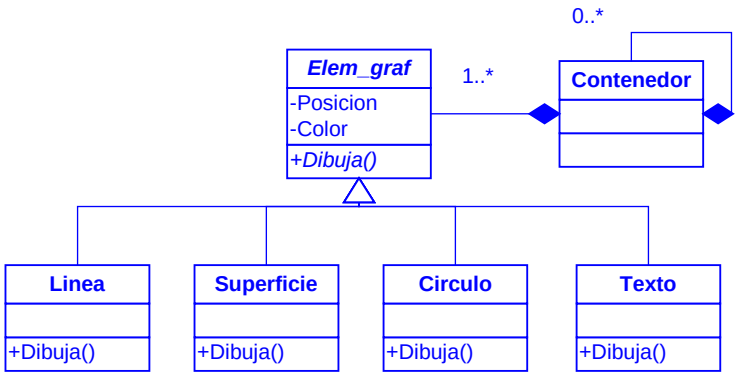
Supóngase ahora un entorno gráfico de ventanas como Windows donde los elementos que se manejan (`Objeto_entorno`) son del tipo `Dibujo`. También aquí hay una serie de elementos simples (como pueden ser `Borde`, `Icono`, `Fondo`, `Etiqueta` o `Decorador`) y otros `Elem_compuesto` son compuestos, como `Ventana`, `Cuadro`, `Marco`, `Barra`, `Menu` o `Boton`. Así, una Barra de Herramientas (`Barra_herr`) es una especialización de `Barra` y está compuesto por un `Borde` y un `Fondo`, por `Botones` y `Etiquetas`, por `Menus` deplegables, por `Separadores` y un `Asa` (estos últimos, los consideramos especializaciones de `Decorador`). Por ejemplo:



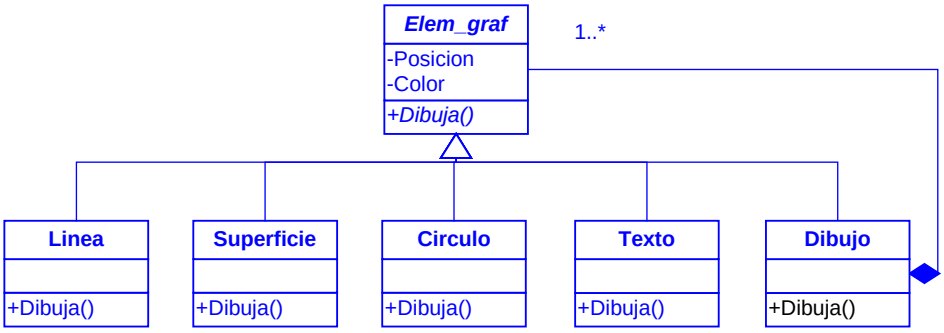
Pregunta c) **Utilice el patrón de diseño 'Composición' anterior para representar la estructura de `Objeto_entorno` y, en concreto, `Barra_herr`.**

Solución

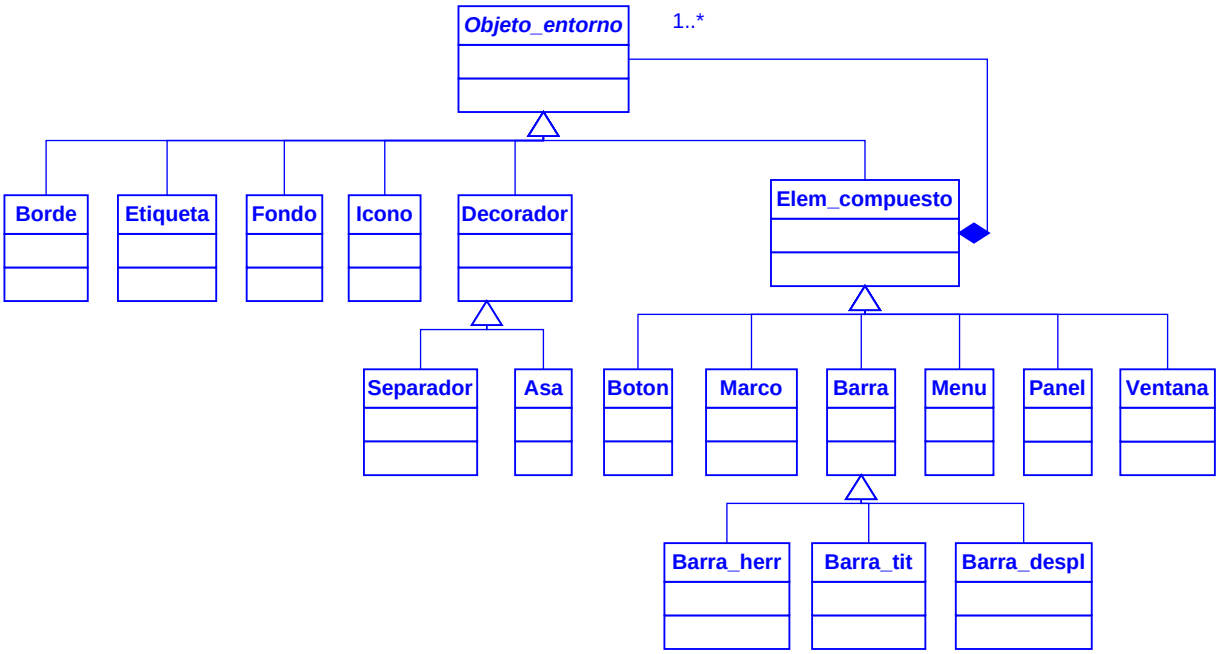
a) Clase Contenedor:



b) Patrón de Composición:



c) Patrón de Composición para Barra_herr:



INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Modelo			
	Junio 2010			Ámbito
				2ª SEMANA Y UE

ESTE EJERCICIO NO ES DE TEST.

NECESITO EL EJERCICIO MANUSCRITO POR EL ALUMNO Y LA HOJA DE LECTURA ÓPTICA PARA PODER CALIFICAR.

SI ESTE EJERCICIO NO SE HA IMPRESO EN HOJA DE LECTURA ÓPTICA, SOLICITE UNA AL TRIBUNAL.

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En cada pregunta teórica, que se valora con 2'5 puntos, la nota mínima es 1 punto; en la segunda parte (ejercicio de teoría aplicada que se valora con 5 puntos) la nota mínima que debe obtener es de 2 puntos.

¡ATENCIÓN! PONGA SUS DATOS EN LA HOJA DE LECTURA ÓPTICA QUE DEBERÁ ENTREGAR JUNTO CON EL RESTO DE LAS RESPUESTAS.

Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm)

PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. La **Línea Base** es un concepto inherente a la *Gestión de la Configuración* del Software que incide en la idea de mantener el control de los elementos de trabajo en su evolución, que necesariamente tienen, durante el desarrollo de un producto Software. Según IEEE 620.12/1990 se define:

   > *Una especificación o producto que se ha revisado formalmente y sobre el que se ha llegado a un acuerdo, y que de ahí en adelante sirve como base (referencia) para un desarrollo posterior y que puede cambiarse solamente a través de procedimientos formales de control de cambios.*

   En cualquier fase del ciclo de vida, un elemento de trabajo se modifica libremente hasta llegar a un estado en el que algún agente participante, con capacidad y autoridad para ello, lo revisa y aprueba formalmente. A partir de ese momento dicho elemento no se puede modificar si no es mediante un protocolo formal de **control de cambios**. Puede haber líneas base en diferentes ámbitos: Planificación, Requisitos, Diseño, Desarrollo, Configuración del Producto... En todos los casos, la línea base confiere visibilidad, estabilidad y control al desarrollo porque identifica situaciones significativas (estados) de los elementos de trabajo (y su historia), permiten la comparación entre unas y otras para hacer un seguimiento del progreso del desarrollo y porque, además, añaden su capacidad para ser recuperadas.

   Supóngase que, durante la fase de análisis de un producto que se gestiona con una **línea base de requisitos**, el Departamento Comercial de su organización o un usuario potencial del producto, solicita que se incorporen al producto determinadas funcionalidades.

   Según lo explicado más arriba y lo que se ve en la asignatura sobre gestión de la configuración, razone cómo se manejaría esta solicitud y su incidencia en la elaboración de los requisitos del sistema. ¿Debería aceptarse la solicitud?

   Solución

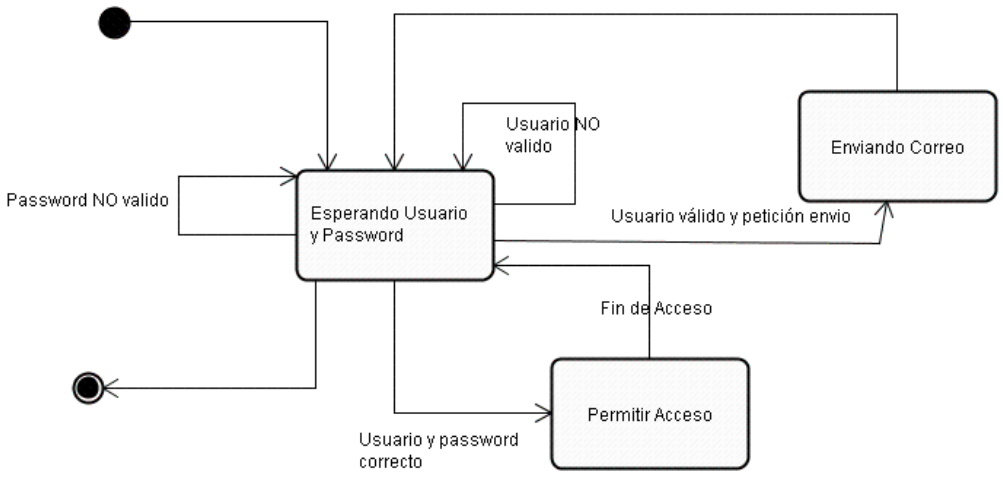
   En la fase de especificación de un producto, es usual que '*lluevan*' requisitos desde distintas procedencias. Este es uno de los motivos por los que, en la asignatura, se hace hincapié en la importancia de identificar la figura del cliente como interlocutor directo en la mayor parte del desarrollo. Sin embargo, esto no quiere

decir que sea el único. Lo más común es que haya que considerar necesidades y aspectos relacionados con el futuro uso de la aplicación o tener en cuenta decisiones estratégicas al respecto de la competitividad del producto en el mercado o de la propia organización. Esto contribuye a la volatilidad de los requisitos, su incertidumbre y disminuye la controlabilidad del desarrollo. Como bromea Kenneth M. Dymond en "*UNA GUÍA DEL CMM: COMPRENDER EL MODELO DE MADUREZ DE CAPACIDAD DEL SOFTWARE*": *«… demasiados cocineros estropean el caldo»*. Precisamente, el concepto de la línea base es paliar estos efectos. En la línea base de requisitos, debería identificarse los interlocutores válidos para incorporar requisitos y cuál es el procedimiento para admitirlos. De esta manera, si la solicitud del enunciado se produce con anterioridad a la consolidación de una línea base de requisitos, debería evaluarse superficialmente y admitirse condicionalmente a lo que establezca la línea base o a lo que se acuerde al constituirla. Otra opción es recoger la solicitud y postergar la decisión para tratarla como un cambio sobre la línea base, una vez constituida. En cualquier caso, tanto la admisión de un agente peticionario de nuevos requisitos como la incorporación de estos requisitos, requiere la evaluación de la conveniencia y el alcance de la modificación, el acuerdo de su incorporación –o no— y la comunicación de dicho acuerdo a todas las partes implicadas.

2. Realice estos apartados:
   a) Modele, mediante un diagrama de transición de estados (DTE), el módulo de acceso a un sistema mediante usuario y contraseña. Contemple la posibilidad de enviar por correo electrónico la contraseña a los usuarios válidos que así lo soliciten en caso de olvido.
   b) Descríbalo también mediante pseudocódigo.

Solución

a)



b)

```
SI (usuario) Y (password) correctos ENTONCES
        Acceso permitido
   SI NO
      SI (usuario) correcto Y peticion de envio ENTONCES
           enviar password por correo electrónico
         SI NO
           Denegar acceso
```

SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Se desea diseñar un sistema informático para controlar la apertura y cierre de la compuerta trasera de un camión. Tal como indica la figura, el sistema recibirá información de tres palancas (P1, P2 y P3) y diversos sensores:

   1) La palanca P1 permite subir y bajar la compuerta en vertical. Un sensor indicará si la compuerta está a ras del suelo (0%) o arriba del todo (100%).

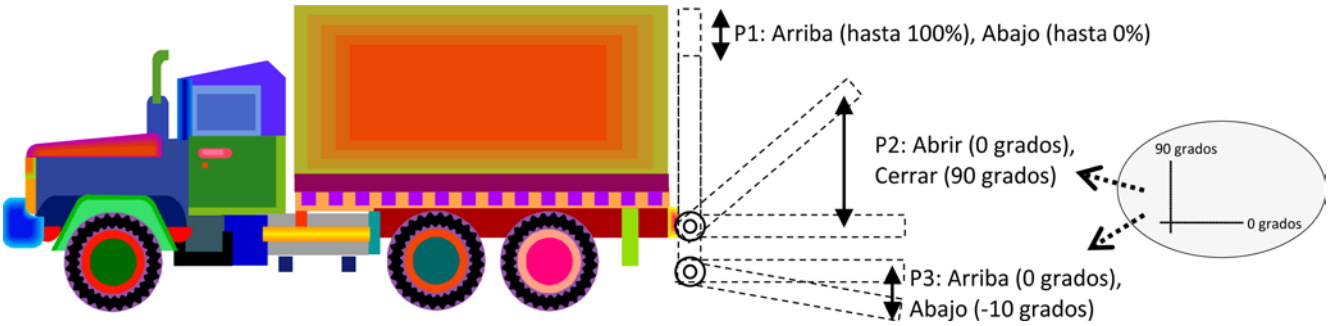
   2) La palanca P2 abre y cierra la compuerta. Un sensor indicará el ángulo de apertura de la compuerta. Se considera que la compuerta está abierta si el ángulo es de 0 grados y cerrada si el ángulo es de 90 grados.

   3) La palanca P3 inclina la compuerta para facilitar la carga y descarga del camión. El ángulo que puede girarse la compuerta con esta palanca va de 0 a -10 grados.

   El proceso de apertura se consigue accionando las palancas: P2 (abrir) → P1 (bajar) → P3 (inclinar)

   El proceso de cierre se consigue accionando las palancas: P3 (arriba) → P1 (subir) → P2 (cerrar)

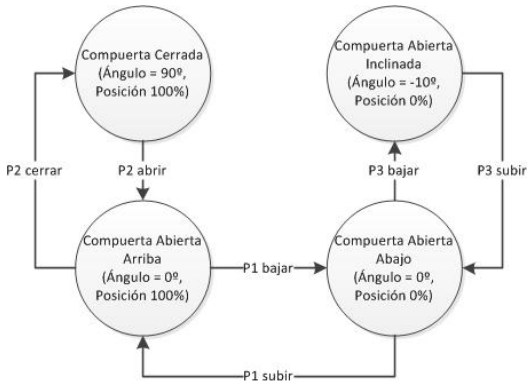
   El sistema informático deberá asegurar que la compuerta sólo se abre o se cierra si está arriba del todo (100% de la posición vertical).



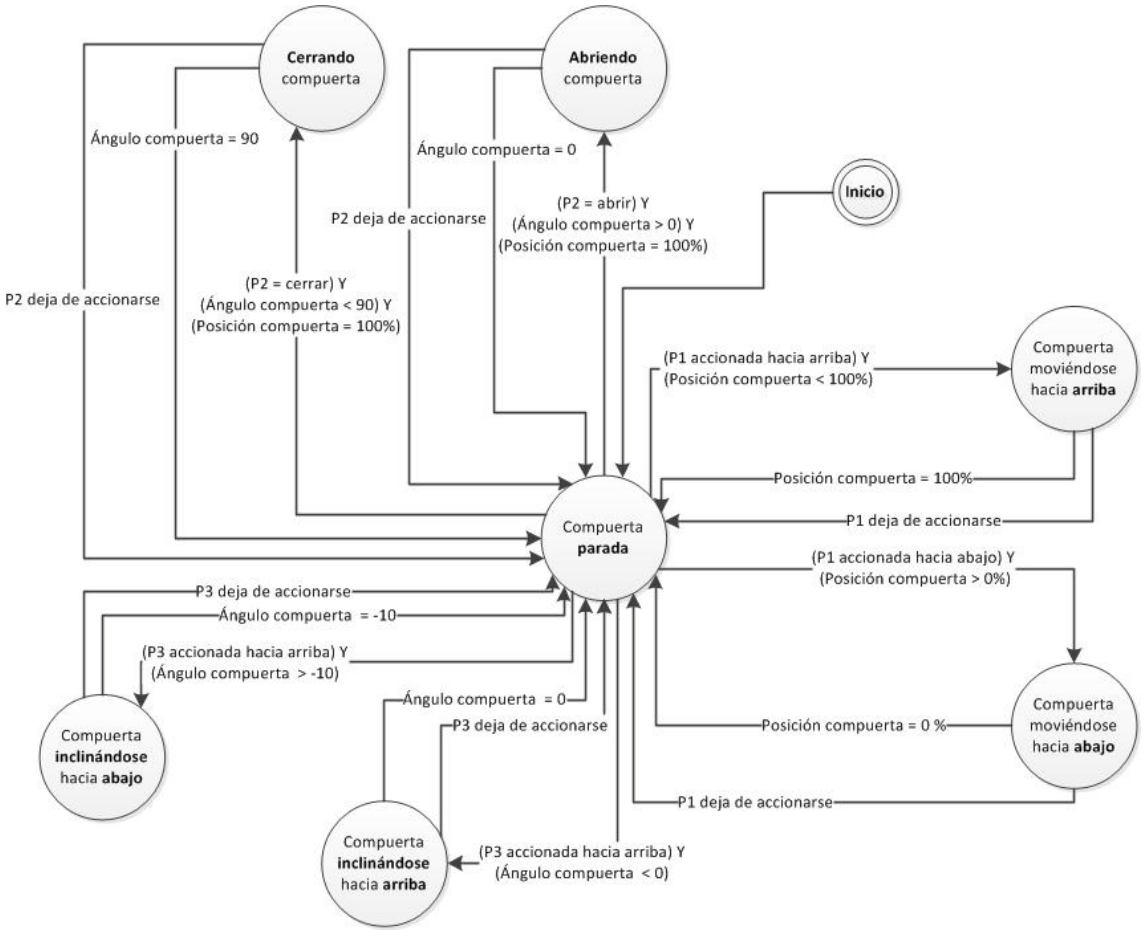
   Diseñe el sistema informático mediante un diagrama de transición de estados. No contemple la posibilidad de accionar más de una palanca a la vez.

Solución

Según el enunciado estricto:



Al considerar que los movimientos de la compuerta son estados, se introducen nuevas posibilidades. Por ejemplo, imaginemos que mientras se está bajando la compuerta, se decide rectificar y volver a subir la compuerta. Esta posibilidad, que no está soportada por la solución anterior, conllevaría la siguiente secuencia de eventos en la solución: P1 accionada hacia abajo -> P1 deja de accionarse -> P2 accionada hacia arriba.



INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Modelo			
	Septiembre 2010			Ámbito
				NACIONAL – UE ORIGINAL

ESTE EJERCICIO ES DE TIPO MIXTO.

ES IRRELEVANTE SI CONTESTA A LA PREGUNTA DE TEST O NO. SIN EMBARGO, SE DEBE ESCANEAR DICHA HOJA JUNTO CON EL RESTO DE LA CONTESTACIÓN DEL EXAMEN. EL ALUMNO PUEDE QUEDARSE CON EL ENUNCIADO.

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En cada pregunta teórica, que se valora con 2'5 puntos, la nota mínima es 1 punto; en la segunda parte (ejercicio de teoría aplicada que se valora con 5 puntos) la nota mínima que debe obtener es de 2 puntos.

Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm)

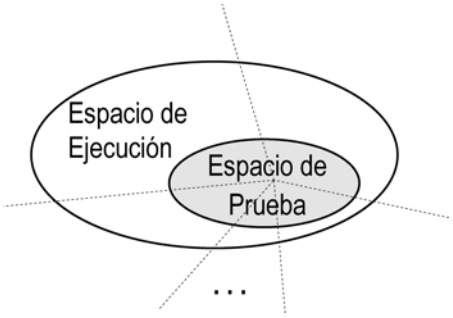
PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

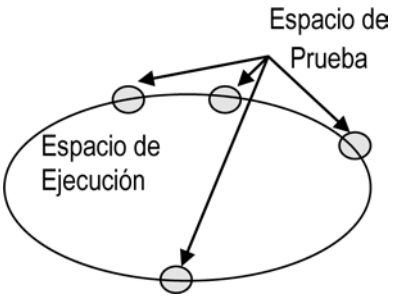
1. Justifique si es recomendable, o no, utilizar un modelo de ciclo de vida evolutivo para el desarrollo de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido o de alto índice de riesgo.

   Solución

   En todos estos sistemas, su naturaleza y funcionalidad dependen fuertemente de la interacción de todos los componentes o de la coordinación en el desarrollo o del funcionamiento de sus elementos. Por ello, es difícil o inviable hacer desarrollos incrementales o parciales, con entornos reducidos del sistema –prototipos—, sin comprometer seriamente el producto final. El uso de este modelo de desarrollo no se recomienda para productos de este tipo.

2. Las siguientes figuras representan dos métodos de prueba de caja negra: el análisis de valores límite y la partición en clases de equivalencia.
   1) Estos métodos limitan el *espacio de prueba* a una parte del *espacio de ejecución*, ¿por qué?
   2) Identifique qué método corresponde a cada figura y resuma brevemente en qué consiste cada método.

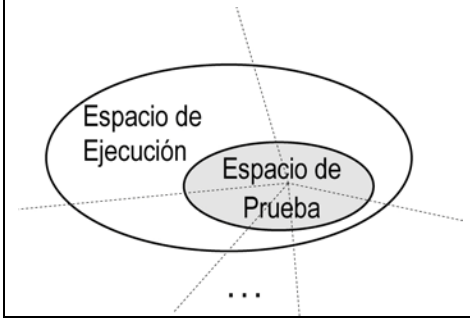


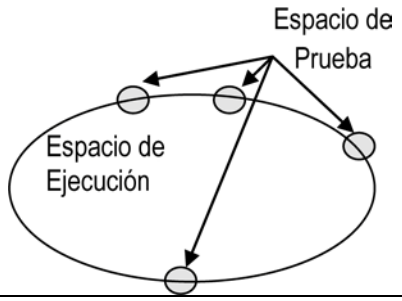


Solución

a) En general, probar completamente un programa es inabordable y además no resulta rentable ni práctico. Por esta razón, se emplean métodos que limitan las pruebas a ciertas regiones del espacio de ejecución. Con las pruebas sólo se explora una parte de todas las posibilidades del programa. Se trata de alcanzar un compromiso para que con el menor esfuerzo posible se puedan detectar el máximo número de defectos y, sobre todo, aquellos que puedan provocar las consecuencias más graves.

b)

Partición en clases de equivalencia	
	Este método divide el espacio de ejecución de un programa en varios subespacios o clases equivalentes. Cada clase (delimitada por un par de líneas punteadas en la figura) agrupa a todos aquellos datos de entrada al programa que producen resultados equivalentes.

Análisis de valores límite	
	Muchos programas se construyen codificando primero un tratamiento general, y retocando luego el código para cubrir casos especiales. Por esta y otras razones es bastante normal que los errores tengan cierta tendencia a aparecer precisamente al operar en las fronteras o valores límite de los datos normales (representados por círculos sombreados en la figura). Este método se basa en la identificación y prueba de los valores límite.

SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Se ha diseñado un paquete de revisión ortográfica mediante refinamiento progresivo –diseño funcional descendente o, también llamado, diseño algorítmico—. La descripción del comportamiento del módulo es la siguiente:

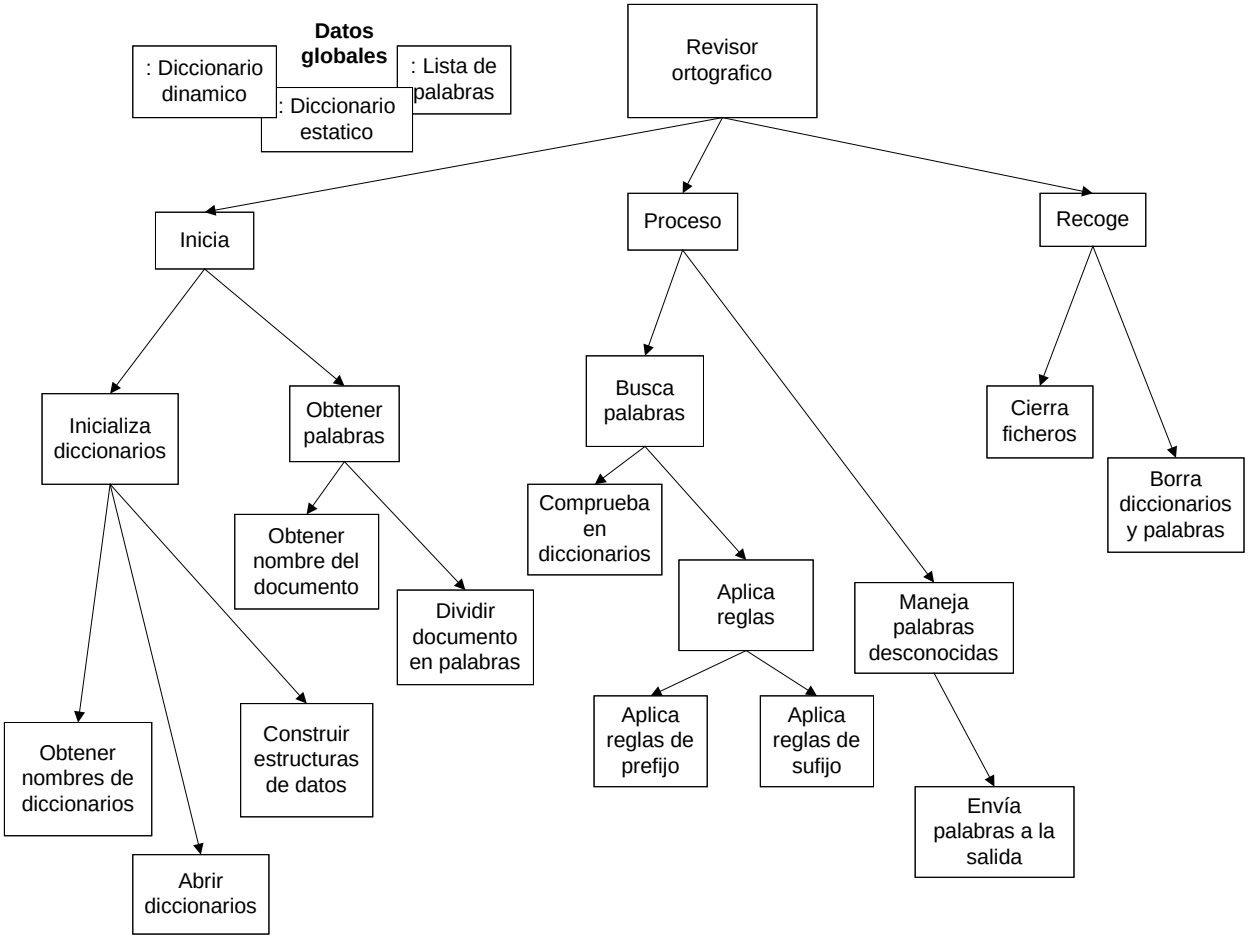
   "Recoge las palabras de un documento de texto particular y las busca, una a una, bien en un diccionario principal –fichero estático, que el usuario no puede modificar— o bien en un diccionario definido por el usuario –dinámico, que sí se puede ampliar o modificar—. Ambos diccionarios son ficheros constituidos por una lista de palabras ordenadas alfabéticamente. Si alguna de las palabras del documento revisado no aparece en ninguno de los diccionarios o no se puede deducir mediante la aplicación de ciertas transformaciones, prefijos o sufijos; se mostrará por la salida estándar del sistema."

   Las especificaciones y objetivos iniciales del programa son:

   - Debe manejar archivos de texto ASCII.
   - El documento debe caber completamente en la memoria principal.
   - Debe ejecutarse "*rápidamente*".

- Tenga en cuenta que el documento se procesa en modo '*batch*', "*por lotes*".
- Debe ser '*inteligente*' sobre lo que constituye las palabras mal escritas (por eso necesitamos prefijo/sufijo, reglas y un diccionario privado, dinámico o de usuario).

El gráfico del diseño obtenido es el siguiente:

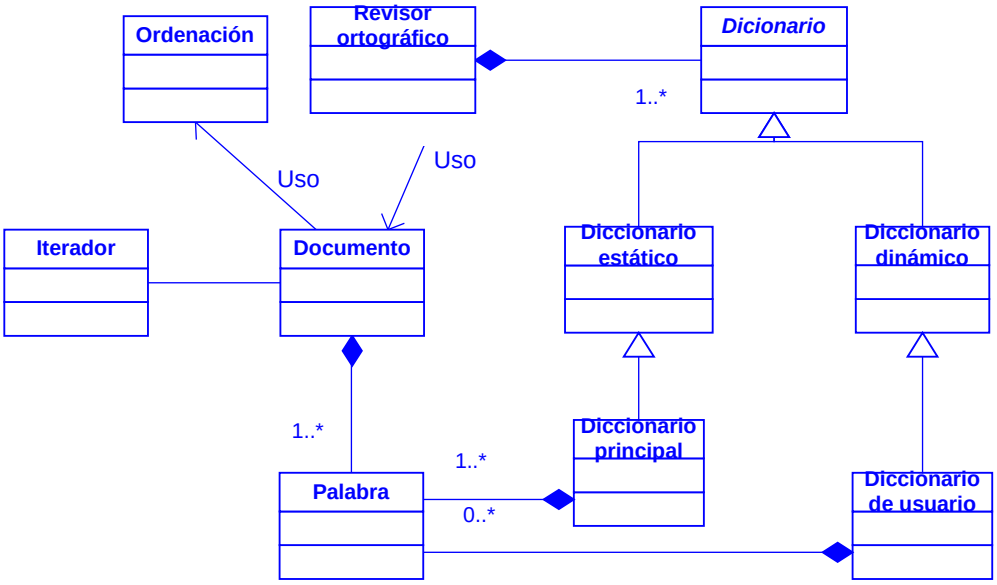


Las desventajas de este diseño son:

- Gestiona con dificultad la evolución (cambios) del sistema a largo plazo.
  - Por ejemplo, los cambios en los algoritmos y estructuras de datos repercuten en la estructura de todo el diseño (y en la documentación…).
  - La implementación se caracteriza, a menudo, por la falta de ocultación; combinada con una sobre-abundancia de variables globales.
    - Estas características no son inherentes aunque, frecuentemente, están relacionadas.
- No facilita la reutilización.
  - El diseño se adapta a medida para los requisitos y especificaciones de una aplicación en particular.
- estructuras de datos.
  - Se aplazan hasta que las funciones y procedimientos se hayan definido y ordenado.

Para paliar los puntos débiles de este diseño, desarrolle otro arquitectura pero con orientación a objetos. Explique claramente cómo construye dicha arquitectura y justifique las decisiones que tome.

Solución


Ordenación
Revisor ortográfico
Dicionario
1..*
Uso
Uso
Iterador
Documento
Diccionario estático
Diccionario dinámico
1..*
1..*
Diccionario principal
Palabra
0..*
Diccionario de usuario

INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Modelo			
	Septiembre 2010			Ámbito
				NACIONAL – UE RESERVA

ESTE EJERCICIO ES DE TIPO MIXTO.

ES IRRELEVANTE SI CONTESTA A LA PREGUNTA DE TEST O NO. SIN EMBARGO, SE DEBE ESCANEAR DICHA HOJA JUNTO CON EL RESTO DE LA CONTESTACIÓN DEL EXAMEN. EL ALUMNO PUEDE QUEDARSE CON EL ENUNCIADO.

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En cada pregunta teórica, que se valora con 2'5 puntos, la nota mínima es 1 punto; en la segunda parte (ejercicio de teoría aplicada que se valora con 5 puntos) la nota mínima que debe obtener es de 2 puntos.

Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm)

PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. Acaba de incorporarse a la empresa *'Victoria & Niagara Software'* como director/a de software. Dicha empresa lleva muchos años desarrollando software de gestión contable para pequeñas empresas y utilizando el ciclo de vida en cascada con éxito aceptable. Sin embargo, según su experiencia, usted piensa que el modelo con prototipo rápido es una forma bastante mejor para desarrollar software. Escriba un informe, dirigido al vicepresidente de desarrollo de software, explicando por qué cree que la organización debería cambiar al uso del modelo con prototipo rápido. Recuerde que a los vicepresidentes no les agradan los informes de más de una página.

   Solución

   La ventaja del ciclo de vida en cascada es que establece un estilo de trabajo disciplinado y está dirigido por la documentación que se va generando. Sin embargo, no garantiza que el producto entregado sea el que necesita el cliente, porque la línea de fabricación se 'separa' del contacto con el cliente a partir de la fase de análisis. Con el modelo con prototipo rápido, sin embargo, el cliente ve inmediatamente si sus necesidades se reflejan, o no, en el prototipo y esto aumenta la confianza en la garantía de que el producto final responda a sus necesidades. Por otro lado, el hecho de que la organización tenga experiencia en un dominio concreto, hace que la creación de un prototipo sea casi inmediata. O dicho de otra manera, la experiencia de la organización permite crear un esquema o patrón general, aplicable al dominio, del que se puede obtener rápidamente (con una parametrización adecuada) el prototipo rápido necesario para cada desarrollo particular.

2. Explique en qué consiste la fase de ***mantenimiento del software*** y distinga entre los distintos tipos de mantenimiento: correctivo, adaptativo y perfectivo.

   Solución

   Pág. 24, sección 1.8.1, del libro de la asignatura.

SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. A principios de los 80 el número de partículas elementales conocidas aumentó muchísimo, con lo que se intentó organizarlas en familias con propiedades comunes. Algunas partículas (como el electrón y el neutrino) no experimentan la interacción fuerte y se las denomina **leptones.** Las partículas del núcleo del átomo experimentan la interacción fuerte y se las conoce como **hadrones**. Los hadrones se subdividen en dos categorías: los mesones (como el pión) y los bariones (como el protón).

   Las partículas poseen un momento angular intrínseco que se conoce como ***spin***, cuya magnitud es un múltiplo de la constante de Plank ***h***. Para los bariones este múltiplo es un semientero: 1/2, 3/2, 5/2, etc. mientras que para los mesones es un entero: 0, 1, 2, etc. Todos los leptones tienen spin 1/2 ***h***.

   Modele la descripción anterior mediante un diagrama Orientado a Objetos.

Solución