



EXÁMENES RESUELTOS

INGENIERÍA DEL SOFTWARE

FORMÁTICA SISTEMAS Y GESTIÓN

EXÁMENES INGENIERÍA DEL SOFTWARE



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN
Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN
NACIONAL 1ª SEMANA



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: 28 de mayo de 2003

Hora: 11:30

Duración: 2 horas

MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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. Deduzca y justifique en qué casos es mejor aplicar un ciclo de vida en cascada respecto a uno en espiral y viceversa.
2. Dé una breve definición de la fase de diseño. Describa cuáles son los objetivos últimos que se pretende alcanzar con la descomposición modular del diseño.

SEGUNDA PARTE. PREGUNTA DE APLICACIÓN (5 PUNTOS)

3. Con el fin de promocionar el uso del transporte público y el ocio al aire libre, RENFE ha decidido encargar la construcción de un sistema informático que asesore a sus clientes acerca de 'rutas verdes' para hacer a pie a partir de sus estaciones de tren.

El sistema recibirá periódicamente la siguiente información:

- Un informe meteorológico del Instituto Nacional de Meteorología que contendrá las previsiones climáticas para los próximos días.
- Datos referentes a las estaciones de tren, horarios y precios de billetes. Esta información será suministrada por RENFE.
- Se ha encargado a la empresa "Viajes Najarra" la elaboración e introducción en el sistema de las rutas verdes. Para ello, la empresa podrá solicitar del sistema un informe de las estaciones de RENFE existentes.

Los clientes introducirán en el sistema sus preferencias. A partir de estas y los datos antes descritos, se construirá un informe con las rutas aconsejadas.

Analice el sistema mediante DFDs (Diagramas de Flujo de Datos), desarrollando exclusivamente los DFDs de nivel 0 y 1.



Universidad Nacional
de Educación a
Distancia

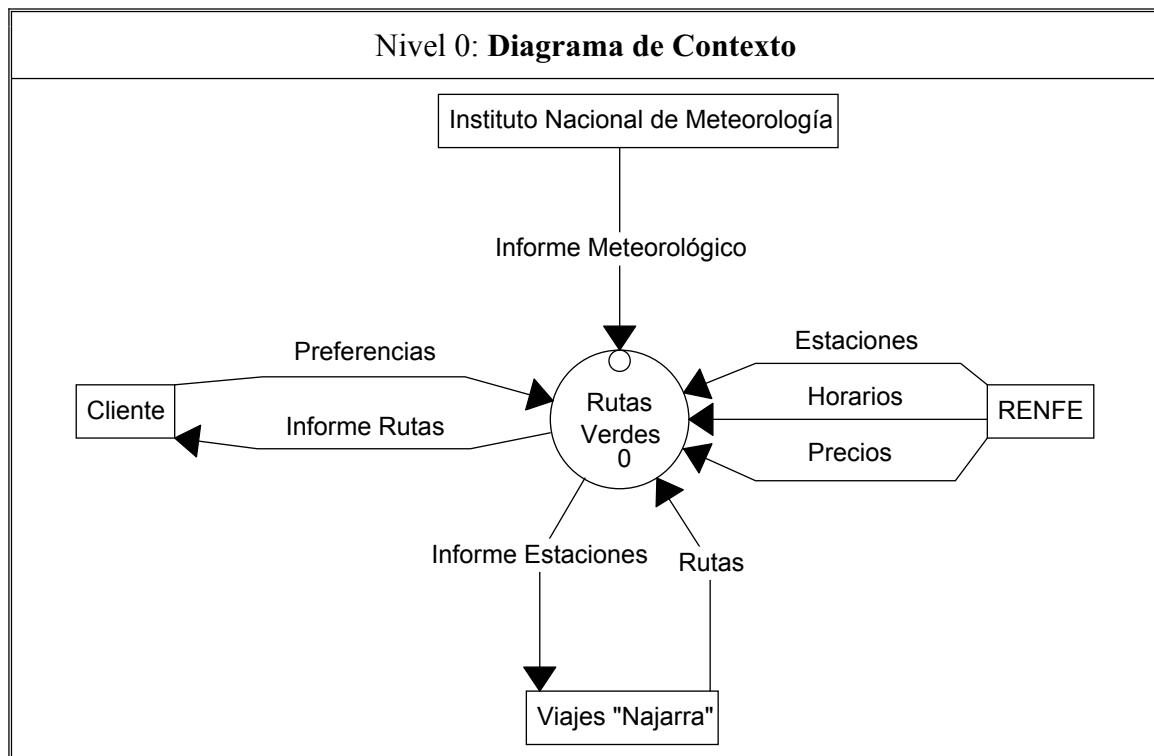
Solución propuesta por el equipo docente

Autores: José Félix Estívariz López, Rubén Heradio Gil, Juan
Antonio Mascarell Estruch

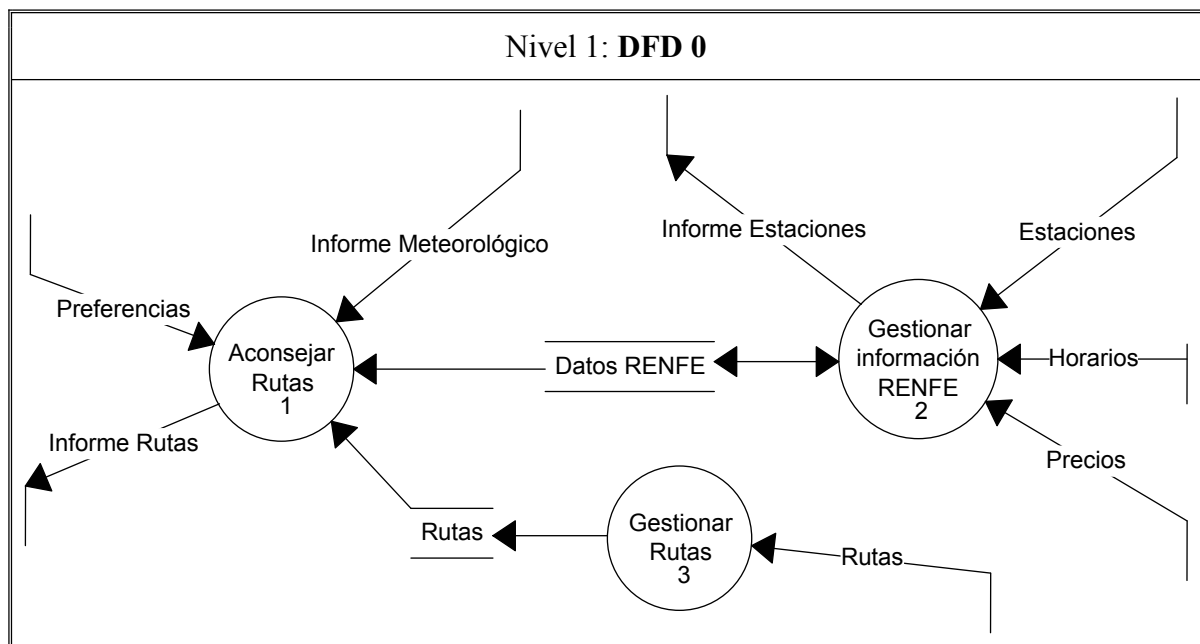


Departamento de
Lenguajes y Sistemas
Informáticos

Nivel 0: Diagrama de Contexto



Nivel 1: DFD 0





UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN
Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN
NACIONAL 2ª SEMANA



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: 11 de junio de 2003

Hora: 11:30

Duración: 2 horas

MATERIAL: NINGUNO

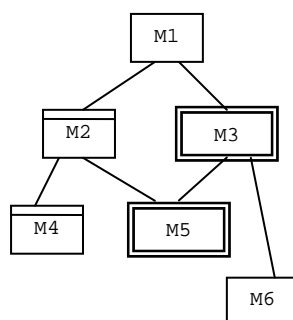
Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) la nota mínima es 1 punto; en la segunda parte (ejercicio de teoría aplicada que se valora 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. Enumere, defina y explique brevemente las fases que componen el ciclo de vida en cascada.
2. Dado el siguiente Diagrama de Abstracciones:



Establezca una comparativa entre las estrategias de integración ascendente y descendente, indicando el orden en el que se construirían los módulos del diagrama para cada una de ellas, así como sus ventajas e inconvenientes.

SEGUNDA PARTE. PREGUNTA DE APLICACIÓN (5 PUNTOS)

3. Se desea construir un sistema que reciba como entrada una lista ordenada de líneas, cada una de las cuales está formada por una lista ordenada de palabras, que a su vez será una lista ordenada de caracteres. Sobre cada línea se pueden realizar rotaciones, que consisten en eliminar la primera palabra y concatenarla al final de la línea. El sistema devolverá como resultado una lista con las posibles rotaciones de todas las líneas ordenadas alfabéticamente (incluyendo las rotaciones nulas).

Por ejemplo,

voy a cenar
al restaurante



a cenar voy
al restaurante
cenar voy a
restaurante al
voy a cenar

Diseñe el sistema utilizando diagramas de abstracciones.



Universidad Nacional
de Educación a
Distancia

Soluciones propuestas por el equipo docente

Autores: José Félix Estívariz López, Rubén Heradio Gil, Juan Antonio Mascarell Estruch

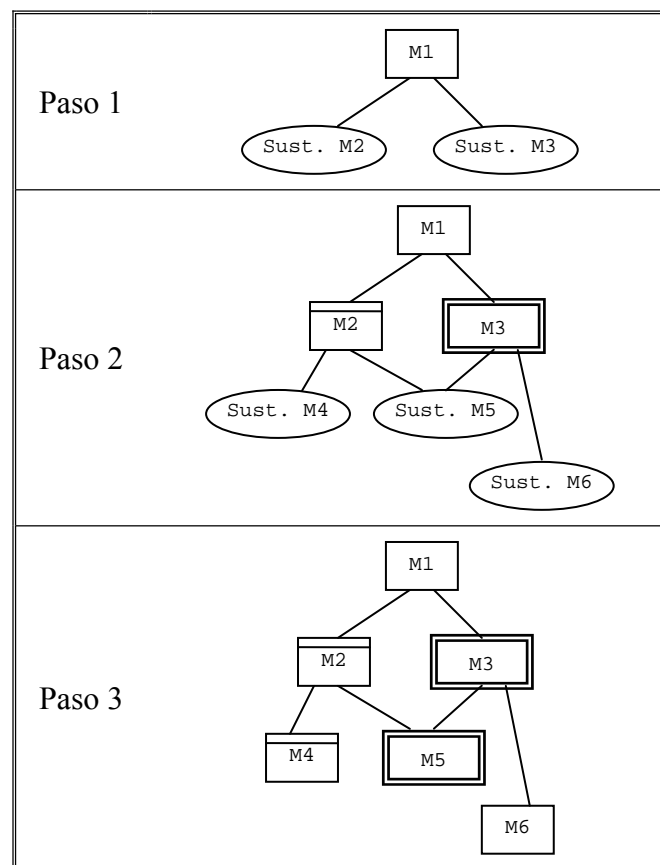


Departamento de
Lenguajes y
Sistemas
Informáticos

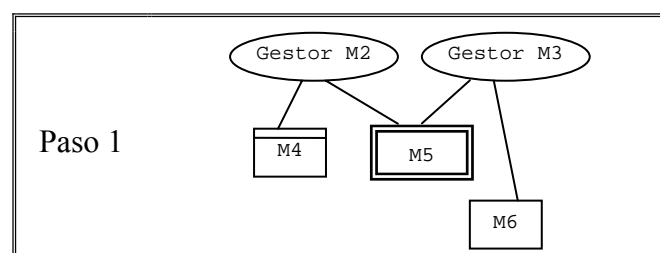
Solución a la pregunta teórica 2

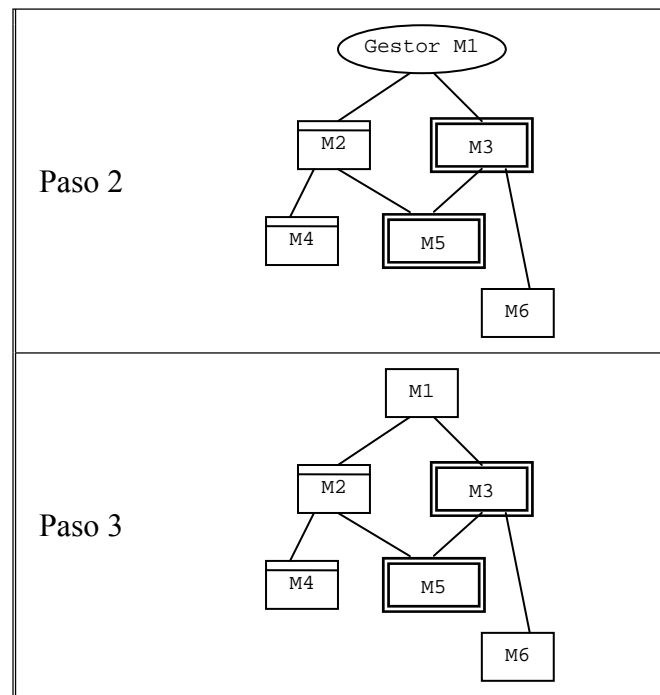
Examinemos el orden de realización de los módulos para cada estrategia de integración:

Integración Descendente



Integración Ascendente





La siguiente tabla resume las ventajas e inconvenientes derivados del uso de las integraciones descendente y ascendente en el ejemplo que nos ocupa:

	I. Descendente	I. Ascendente
¿Facilita una visión general de la aplicación desde el principio?	Sí	No
Número de elementos de código desechables a construir	5 sustitutos o stubs	3 gestores o drivers
¿Facilita el ensayo de situaciones especiales para los módulos?	No	Sí
¿Facilita el trabajo en paralelo?*	Sí	Sí

(*): Aunque generalmente la integración ascendente propicia en mayor grado el trabajo en paralelo que la integración descendente, en este ejemplo, las dos estrategias de integración facilitan el trabajo en paralelo en el mismo grado.

Solución a la pregunta de aplicación

En una primera aproximación, aplicaremos el método de Abbott para determinar las abstracciones que componen el diseño. Para ello, marcaremos en rojo los sustantivos (candidatos a ser tipos de datos) y en azul los verbos (candidatos a ser operaciones).

*Se desea construir un sistema que reciba como entrada una **lista ordenada de líneas**, cada una de las cuales está formada por una **lista ordenada de palabras**, que a su vez será una **lista ordenada de caracteres**. Sobre cada línea se pueden **realizar rotaciones**, que consisten en **eliminar la primera palabra** y **concatenarla** al final de la línea. El sistema devolverá como resultado una **lista con las posibles rotaciones de todas las líneas ordenadas alfabéticamente** (incluyendo las rotaciones nulas).*

A partir de este marcado elaboramos una doble lista con los elementos correspondientes a datos y a operaciones.

DATOS	OPERACIONES
<ul style="list-style-type: none"> • lista ordenada de líneas • lista ordenada de palabras • lista ordenada de caracteres • lista con las posibles rotaciones de todas las líneas ordenadas alfabéticamente 	<ul style="list-style-type: none"> • realizar rotaciones • eliminar la primera palabra • concatenar la primera palabra al final de la línea

¿Qué son los Tipos Abstractos de Datos?

Un Tipo Abstracto de Datos (TAD) es una entidad que agrupa la estructura de un tipo de datos con las operaciones necesarias para su manejo.

Tradicionalmente, en la programación imperativa se optaba por separar los programas en dos partes: la de proceso y la de datos (ejemplos de lenguajes imperativos son FORTRAN, COBOL, PASCAL, BASIC...). La parte de proceso accedía y operaba directamente sobre los datos.


Esta separación produce una independencia funcional muy baja. Un ejemplo que ilustra esto es el “efecto 2000”, donde la simple adición de dígitos al formato de las fechas supuso realizar muchas modificaciones en la parte de proceso.


Para solventar estos problemas surgió el concepto de Tipo Abstracto de Datos, que junta la representación de los datos con la parte de proceso que los manipula. Los TADs ocultan la representación de los datos, que sólo es accesible desde las operaciones. De esta forma, un cambio en la representación de los datos de un TAD no se propaga a todo el programa, sino solamente a las operaciones del TAD.



Antiguamente, las estructuras de datos se venían definiendo por su representación. Sin embargo, el paso clave hacia la abstracción de los datos es invertir este punto de vista: olvidar por el momento la representación y considerar que las operaciones en sí mismas definen la estructura de datos.

NOTA: *un error grave cometido por muchos alumnos consiste en diseñar una operación fuera del TAD que le corresponde. Por ejemplo, en este examen algunos alumnos proponían utilizar un TAD llamado “Rotaciones” y una abstracción funcional separada llamada “ProducirRotaciones”.*

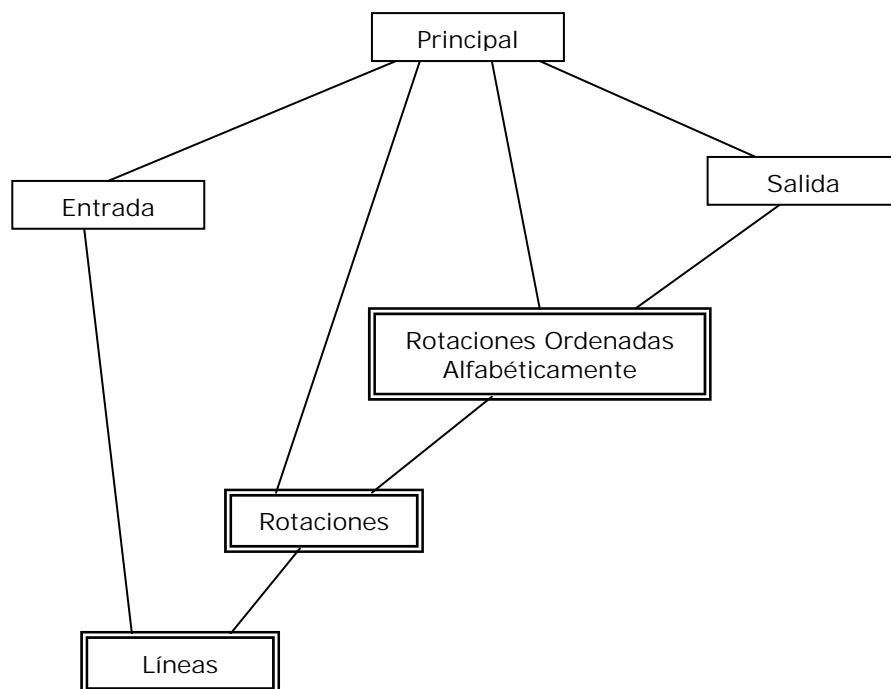
Tras analizar la doble lista sobre datos y operaciones se proponen tres datos encapsulados:

 Los TAD propuestos son datos encapsulados por que sólo existirá una variable o elemento de cada tipo en el sistema.

DATO: Líneas Operaciones: Introducir Obtener	 Se ha tomado la decisión de aglutinar los datos “lista ordenada de líneas”, “lista ordenada de palabras” y “lista ordenada de caracteres” en un sólo dato encapsulado, ya que el
--	--


EstaVacia	<p><i>crecimiento innecesario del número de abstracciones aumentaría la complejidad del sistema. Ésta abstracción consta de tres operaciones:</i></p> <ul style="list-style-type: none"> • “Introducir”: se encargará de recibir una línea y almacenarla en el dato encapsulado. • “Obtener”: proporcionará una línea. • “EstaVacia”: indicará si hay alguna línea almacenada dentro del dato encapsulado.
DATO: Rotaciones Operaciones: ProducirRotaciones Obtener EstaVacia	<p> “Rotaciones” es un dato encapsulado que producirá y almacenará todas las posibles rotaciones realizadas sobre las líneas que el sistema reciba como entrada. La operación “ProducirRotaciones” engloba a las operaciones “realizar rotaciones”, “eliminar la primera palabra” y “concatenar la primera palabra al final de la línea” que aparecen en la doble lista de datos y operaciones.</p>
DATO: RotacionesOrdenadasAlfabéticamente Operaciones: Ordenar Obtener EstaVacia	<p> Éste es un dato encapsulado que producirá y almacenará las rotaciones realizadas sobre las líneas ordenadas alfabéticamente.</p>

El diagrama de abstracciones propuesto para construir el sistema solicitado es el siguiente:



Como puede verse, el diseño consta de la abstracción funcional “Principal” que se encargará de coordinar al resto de las abstracciones. Además, el diseño posee las siguientes abstracciones funcionales:

- “Entrada”: se encargará de obtener los datos del exterior (en el enunciado no se especifica la forma en que el usuario introducirá los datos) y de almacenarlos en “Lineas”. Para ello, empleará la operación “Introducir” proporcionada por dicho dato encapsulado.
- “Salida”: mostrará el resultado final de procesar los datos de entrada. Para este fin, utilizará la operación “Obtener” del dato encapsulado “RotacionesOrdenadasAlfabéticamente”.

 Uno de los objetivos de la fase de diseño es facilitar el mantenimiento. El diseño propuesto consigue este objetivo en gran medida. Piense el lector en posibles modificaciones que pueden producirse sobre las especificaciones del sistema y verá que la parte del diseño afectada es mínima y está claramente localizada. A modo de ejemplo, la siguiente tabla propone algunas modificaciones y muestra las partes del sistema que se verían afectadas:

<i>Modificación</i>	<i>Partes afectadas</i>
<i>Presentar el resultado de la ordenación en un nuevo formato gráfico</i>	<i>“Salida”</i>
<i>La ordenación de las rotaciones deja de ser alfabética y pasa a ser de otro tipo</i>	<i>“RotacionesOrdenadasAlfabéticamente”</i>
<i>Las rotaciones se efectuarán de una nueva manera</i>	<i>“Rotaciones”</i>



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN
Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN
EXTRANJERO ORIGINAL



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: 11 de junio de 2003

Hora: 11:30

Duración: 2 horas

MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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. Deduzca y justifique en qué casos es mejor aplicar un ciclo de vida en cascada respecto a uno en espiral y viceversa.
2. Explique la relevancia en el diseño de la cohesión y del grado de acoplamiento entre módulos.

SEGUNDA PARTE. PREGUNTA DE APLICACIÓN (5 PUNTOS)

3. Se desea construir un sistema que reciba como entrada una lista ordenada de líneas, cada una de las cuales está formada por una lista ordenada de palabras, que a su vez será una lista ordenada de caracteres. Sobre cada línea se pueden realizar rotaciones, que consisten en eliminar la primera palabra y concatenarla al final de la línea. El sistema devolverá como resultado una lista con las posibles rotaciones de todas las líneas ordenadas alfabéticamente (incluyendo las rotaciones nulas).

Por ejemplo,

voy a cenar
al restaurante



a cenar voy
al restaurante
cenar voy a
restaurante al
voy a cenar

Diseñe el sistema utilizando diagramas de abstracciones.



Universidad Nacional
de Educación a
Distancia

Solución propuesta por el equipo docente

Autores: José Félix Estívariz López, Rubén Heradio Gil, Juan
Antonio Mascarell Estruch



Departamento de
Lenguajes y
Sistemas
Informáticos

En una primera aproximación, aplicaremos el método de Abbott para determinar las abstracciones que componen el diseño. Para ello, marcaremos en rojo los sustantivos (candidatos a ser tipos de datos) y en azul los verbos (candidatos a ser operaciones).

*Se desea construir un sistema que reciba como entrada una **lista ordenada de líneas**, cada una de las cuales está formada por una **lista ordenada de palabras**, que a su vez será una **lista ordenada de caracteres**. Sobre cada línea se pueden **realizar rotaciones**, que consisten en **eliminar la primera palabra** y **concatenarla** al final de la línea. El sistema devolverá como resultado una **lista con las posibles rotaciones de todas las líneas ordenadas alfabéticamente** (incluyendo las rotaciones nulas).*

A partir de este marcado elaboramos una doble lista con los elementos correspondientes a datos y a operaciones.

DATOS	OPERACIONES
<ul style="list-style-type: none">• lista ordenada de líneas• lista ordenada de palabras• lista ordenada de caracteres• lista con las posibles rotaciones de todas las líneas ordenadas alfabéticamente	<ul style="list-style-type: none">• realizar rotaciones• eliminar la primera palabra• concatenar la primera palabra al final de la línea



¿Qué son los Tipos Abstractos de Datos?

Un Tipo Abstracto de Datos (TAD) es una entidad que agrupa la estructura de un tipo de datos con las operaciones necesarias para su manejo.

Tradicionalmente, en la programación imperativa se optaba por separar los programas en dos partes: la de proceso y la de datos (ejemplos de lenguajes imperativos son FORTRAN, COBOL, PASCAL, BASIC...). La parte de proceso accedía y operaba directamente sobre los datos.

Esta separación produce una independencia funcional muy baja. Un ejemplo que ilustra esto es el “efecto 2000”, donde la simple adición de dígitos al formato de las fechas supuso realizar muchas modificaciones en la parte de proceso.


Para solventar estos problemas surgió el concepto de Tipo Abstracto de Datos, que junta la representación de los datos con la parte de proceso que los manipula. Los TADs ocultan la representación de los datos, que sólo es accesible desde las operaciones. De esta forma, un cambio en la representación de los datos de un TAD no se propaga a todo el programa, sino solamente a las operaciones del TAD.




Antiguamente, las estructuras de datos se venían definiendo por su representación. Sin embargo, el paso clave hacia la abstracción de los datos es invertir este punto de vista: olvidar por el momento la representación y considerar que las operaciones en sí

mismas definen la estructura de datos.

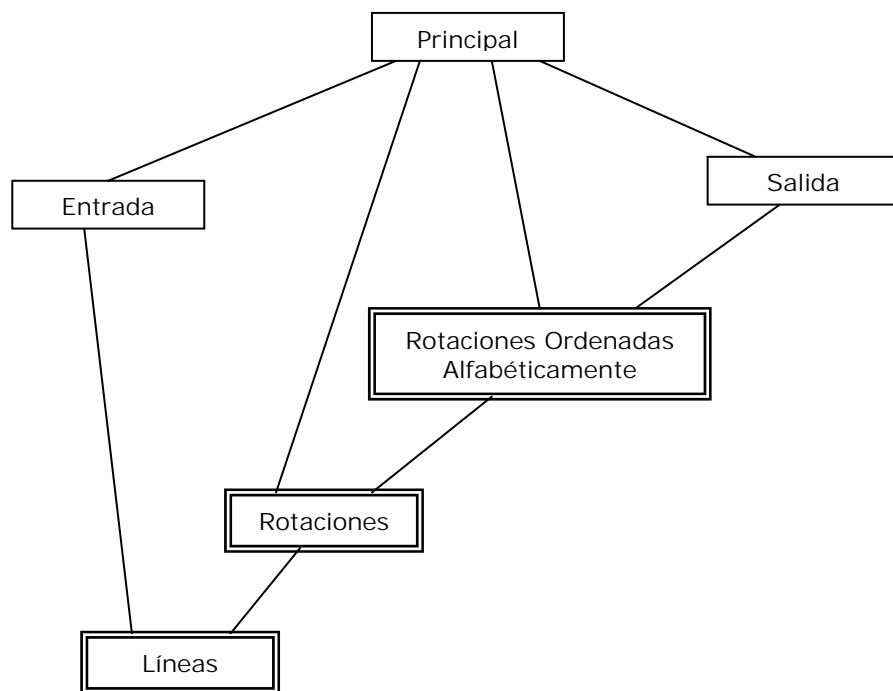
***NOTA:** un error grave cometido por muchos alumnos consiste en diseñar una operación fuera del TAD que le corresponde. Por ejemplo, en este examen algunos alumnos proponían utilizar un TAD llamado “Rotaciones” y una abstracción funcional separada llamada “ProducirRotaciones”.*

Tras analizar la doble lista sobre datos y operaciones se proponen tres datos encapsulados:

 *Los TAD propuestos son datos encapsulados por que sólo existirá una variable o elemento de cada tipo en el sistema.*

DATO: Lineas Operaciones: Introducir Obtener EstaVacia	 <i>Se ha tomado la decisión de aglutinar los datos “lista ordenada de líneas”, “lista ordenada de palabras” y “lista ordenada de caracteres” en un sólo dato encapsulado, ya que el crecimiento innecesario del número de abstracciones aumentaría la complejidad del sistema. Ésta abstracción consta de tres operaciones:</i> <ul style="list-style-type: none">• “Introducir”: se encargará de recibir una línea y almacenarla en el dato encapsulado.• “Obtener”: proporcionará una línea.• “EstaVacia”: indicará si hay alguna línea almacenada dentro del dato encapsulado.
DATO: Rotaciones Operaciones: ProducirRotaciones Obtener EstaVacia	 <i>“Rotaciones” es un dato encapsulado que producirá y almacenará todas las posibles rotaciones realizadas sobre las líneas que el sistema reciba como entrada. La operación “ProducirRotaciones” engloba a las operaciones “realizar rotaciones”, “eliminar la primera palabra” y “concatenar la primera palabra al final de la línea” que aparecen en la doble lista de datos y operaciones.</i>
DATO: RotacionesOrdenadasAlfabéticamente Operaciones: Ordenar Obtener EstaVacia	 <i>Éste es un dato encapsulado que producirá y almacenará las rotaciones realizadas sobre las líneas ordenadas alfabéticamente.</i>

El diagrama de abstracciones propuesto para construir el sistema solicitado es el siguiente:



Como puede verse, el diseño consta de la abstracción funcional “Principal” que se encargará de coordinar al resto de las abstracciones. Además, el diseño posee las siguientes abstracciones funcionales:

- “Entrada”: se encargará de obtener los datos del exterior (en el enunciado no se especifica la forma en que el usuario introducirá los datos) y de almacenarlos en “Líneas”. Para ello, empleará la operación “Introducir” proporcionada por dicho dato encapsulado.
- “Salida”: mostrará el resultado final de procesar los datos de entrada. Para este fin, utilizará la operación “Obtener” del dato encapsulado “RotacionesOrdenadasAlfabéticamente”.



Uno de los objetivos de la fase de diseño es facilitar el mantenimiento. El diseño propuesto consigue este objetivo en gran medida. Piense el lector en posibles modificaciones que pueden producirse sobre las especificaciones del sistema y verá que la parte del diseño afectada es mínima y está claramente localizada. A modo de ejemplo, la siguiente tabla propone algunas modificaciones y muestra las partes del sistema que se verían afectadas:

<i>Modificación</i>	<i>Partes afectadas</i>
<i>Presentar el resultado de la ordenación en un nuevo formato gráfico</i>	<i>“Salida”</i>
<i>La ordenación de las rotaciones deja de ser alfabética y pasa a ser de otro tipo</i>	<i>“RotacionesOrdenadasAlfabéticamente”</i>
<i>Las rotaciones se efectuarán de una nueva manera</i>	<i>“Rotaciones”</i>



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN
Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN
EXTRANJERO RESERVA



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: 14 de junio de 2003

Hora:

Duración: 2 horas

MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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. Explique la diferencia entre las pruebas de **caja negra** y las pruebas de **caja transparente**. ¿Alguna de ellas garantiza la ausencia de fallos?
2. Dé una breve definición de Defecto, Fallo y Error. Explique la diferencia entre la estrategia de prevención de errores frente a la estrategia de recuperación de errores.

SEGUNDA PARTE. PREGUNTA DE APLICACIÓN (5 PUNTOS)

3. En un centro hospitalario se desea informatizar parte de la gestión relativa a sus pacientes. El sistema a construir deberá contemplar las siguientes cuestiones:
 - Un paciente estará asignado a una cama determinada de una planta del hospital, pudiendo estar a lo largo del tiempo de ingreso en diferentes camas y plantas.
 - Para cada paciente se entregarán hasta un máximo de 4 tarjetas de visita. Estas tarjetas servirán para que familiares y amigos del paciente le visiten durante su convalecencia.
 - A un paciente le pueden atender diferentes médicos.
 - Un paciente puede tener distintos diagnósticos de enfermedad.
 - Un médico puede tratar diferentes diagnósticos y viceversa.

Analice el sistema mediante la notación DER (Diagrama Entidad Relación).



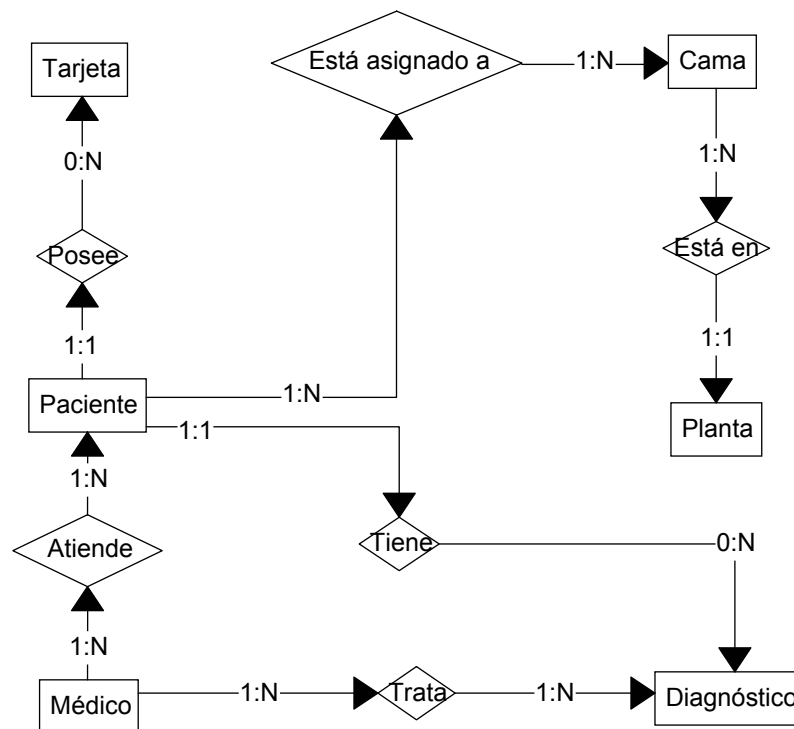
Universidad Nacional
de Educación a
Distancia

Solución propuesta por el equipo docente

Autores: José Félix Estívariz López, Rubén Heradio Gil, Juan
Antonio Mascarell Estruch



Departamento de
Lenguajes y Sistemas
Informáticos



Observación: El anterior diagrama se ha elaborado con la herramienta DOME. La siguiente tabla resume la equivalencia entre la notación de dicha herramienta y la propuesta en el libro de la asignatura.

Notación libro	Notación DOME
d	0:1
	1:1
∞	0:N
K	1:N



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN
Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN
ORIGINAL NACIONAL



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: 4 de septiembre de 2003 Hora: 11:30 Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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. Explique los inconvenientes que tiene el utilizar un ciclo de vida clásico para desarrollar determinado producto en el seno de una organización que carece de experiencia en el desarrollo de este tipo de productos y en las tecnologías implicadas.
2. Supóngase que, para describir el funcionamiento de una aplicación en el diseño, se utiliza un sistema de representación de la realidad con dos categorías: entidades y relaciones. Sitúe, en la categoría que le corresponda, cada uno de los elementos que se manejan en el diseño con abstracciones y en el diseño con objetos.

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

3. Modele el establecimiento y la finalización de una llamada telefónica mediante un Diagrama de Transición de Estados. A continuación, se resume el funcionamiento que debe modelar:
Para establecer una llamada, en primer lugar, se debe descolgar el teléfono. En caso de que no se oiga una señal o tono, se deberá colgar el teléfono y repetir el intento. En caso contrario, se procede a marcar el número de teléfono del destinatario de la llamada. Si el número no es válido, habrá que colgar el teléfono y repetir el proceso desde el principio. Lo mismo ocurre si pasa demasiado tiempo hasta que se marca el número. Si no están ocupadas ni la línea ni la centralita, se establecerá la conexión en cuanto el destinatario descuelgue su teléfono. Por último, la llamada finalizará exclusivamente cuando la persona que inició la llamada cuelgue su teléfono.



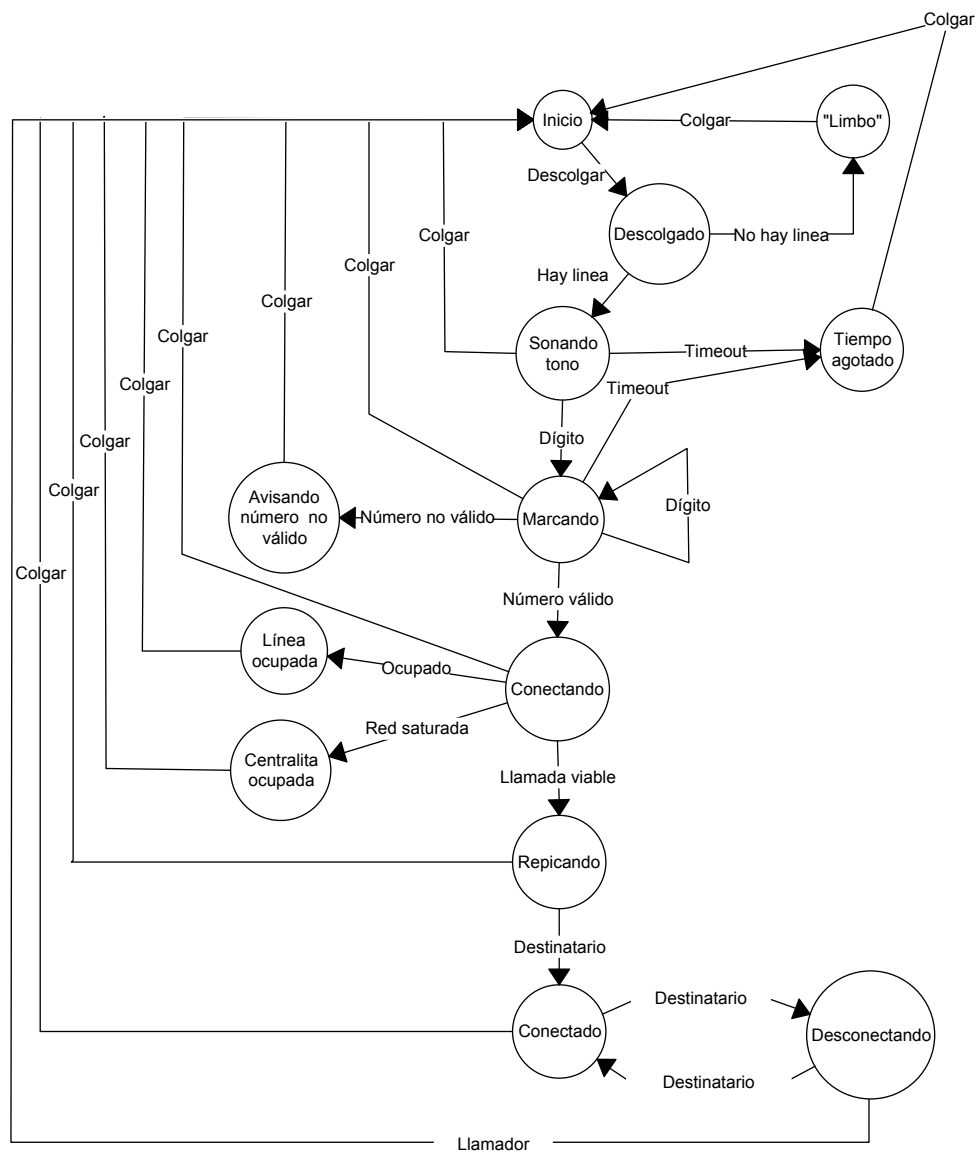
Universidad Nacional de
Educación a Distancia

Solución propuesta por el equipo docente

Autores: José Félix Estívariz López, Rubén Heradio Gil,
Juan Antonio Mascarell Estruch



Departamento de Lenguajes y
Sistemas Informáticos





UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN
Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN
RESERVA NACIONAL



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: 8 de septiembre de 2003 Hora: 16:00 Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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. Explique la diferencia entre especificaciones funcionales y no funcionales en el análisis.
2. El objetivo del diseño es describir el funcionamiento de la aplicación de forma que: la codificación sea fácil y con economía de recursos; el producto funcione según especificaciones y, sobre todo, el mantenimiento sea sencillo. En este sentido, ¿qué inconvenientes acarrea el hacer un diseño de tipo monolítico?

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

3. Se va a construir una estación meteorológica automática junto a un río. Esta estación medirá datos atmosféricos así como niveles de contaminación del río y los transmitirá, vía satélite, a la central de datos. Las especificaciones de funcionamiento son estas:
La temperatura se mide a través de un termopar, estas medidas se realizan cada minuto. Cada 10 minutos se hace la media de las temperaturas leídas y se almacena su valor. Los datos de la presión se leen cada cuarto de hora y se calcula y guarda su media cada hora. También cada hora, se analizan 3 parámetros de nivel de contaminación de las aguas y se registran sus valores. Si algún parámetro pasa cierto umbral de peligro se genera una señal de alarma y se envía automáticamente a la central. Así mismo se mide el caudal del río cada 2 horas. Si se produce una crecida de forma brusca se envía una señal de alarma.
Cada 2 horas la estación automática recopila sus datos los transmite a la central vía satélite. Para ello, previamente tiene que codificar dichos datos en un formato estándar de control de errores para realizar transmisiones tolerantes a fallos.

SE PIDE realizar un Diagrama de Flujos de Datos que modele el sistema anterior.

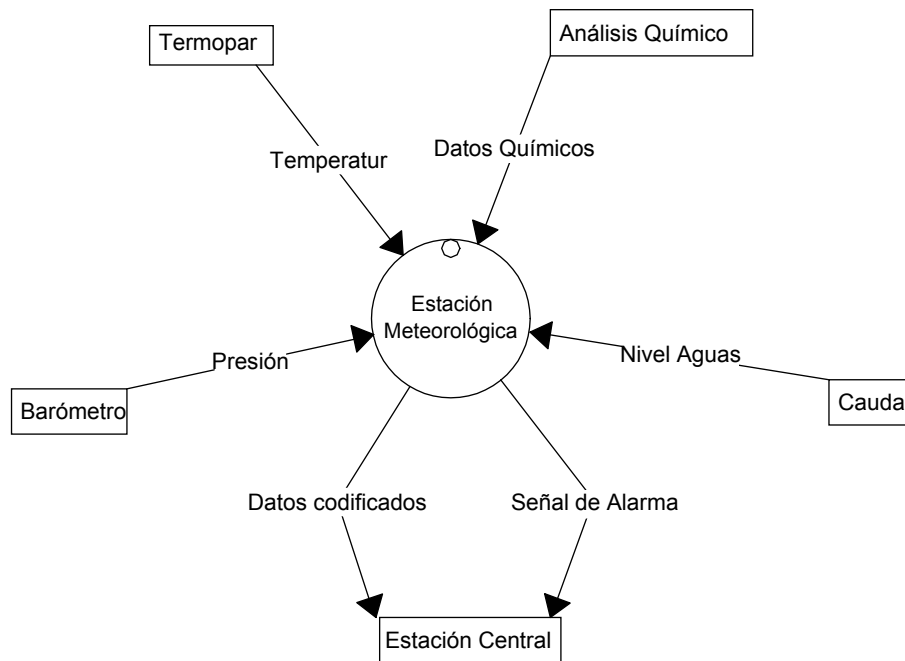


Solución propuesta por el equipo docente

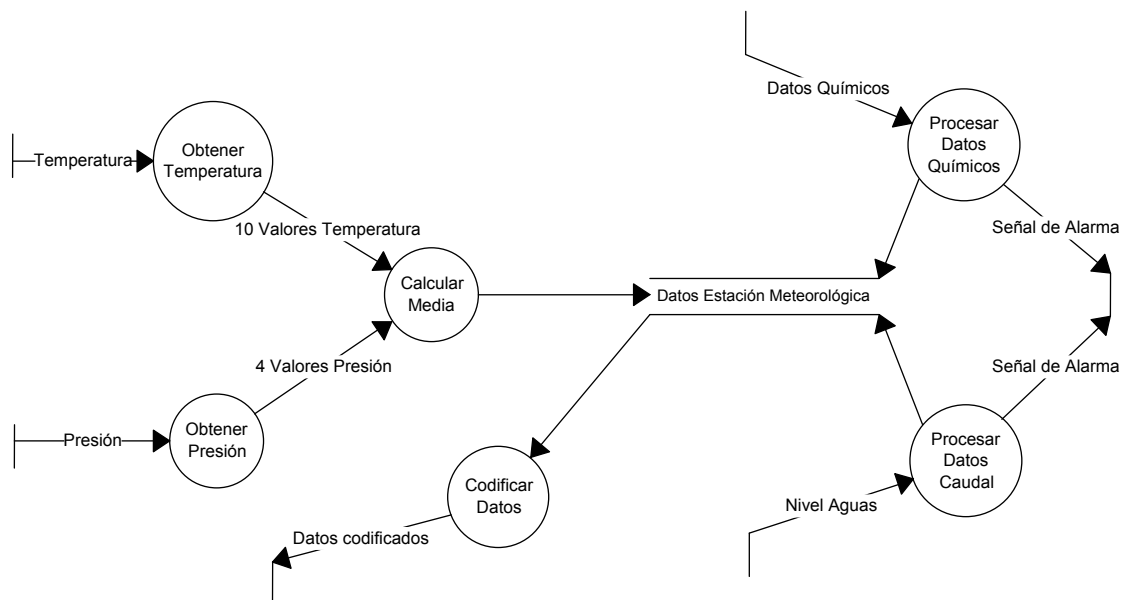
Autores: José Félix Estívariz López, Rubén Heradio Gil,
Juan Antonio Mascarell Estruch



Diagrama de Contexto:



DFD 0:



Observación: dada la simplicidad del DFD 0, se ha optado por no explotarlo en más niveles.



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN
Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN
EXTRANJERO ORIGINAL



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: septiembre 2003

Hora:

Duración: 2 horas

MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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. Suponga que se quiere modelar un sistema en el que lo más importante es representar las tareas y transformaciones que se hacen con la información y los demás aspectos son prácticamente irrelevantes. ¿Sería imprescindible el uso de Diagramas Entidad-Relación? ¿Qué aspecto del modelo prioriza esta notación?
2. ¿El diseño estructurado garantiza, por sí sólo, la independencia funcional? Indique cómo se mide la independencia funcional y explique en qué consisten estas métricas.

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

3. Se nos pide que realicemos la aplicación informática de un cajero automático de un videoclub. El cliente nos da las siguientes especificaciones:

Para entrar al sistema el usuario necesita introducir su tarjeta personal. Lo primero que hace el sistema es comprobar el saldo de la tarjeta. Si tiene saldo cero o negativo solamente permite la acción de recargar tarjeta.

A continuación aparece un menú con las tres únicas opciones: devolver una película, alquilar hasta un máximo de tres películas o recargar la tarjeta.

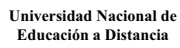
Para alquilar una película la tarjeta tiene que estar actualizada y con saldo. En caso contrario, no permite alquilar. Si la tarjeta tiene saldo, el usuario puede seleccionar hasta un máximo de tres películas siempre y cuando estén disponibles. Para retirarlas debe proceder a confirmar los títulos elegidos.

Al devolver la película el sistema calcula el importe y actualiza la tarjeta.

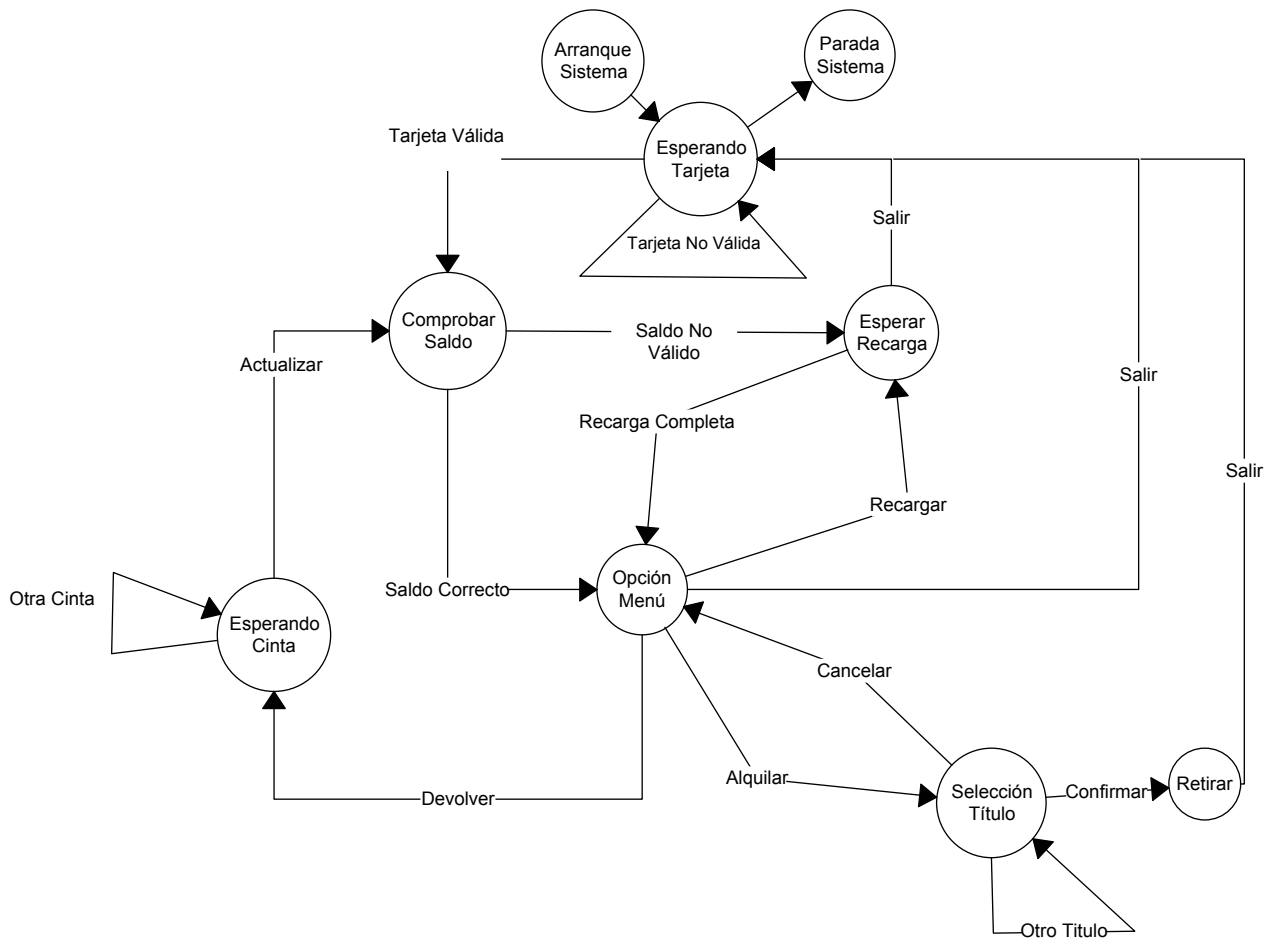
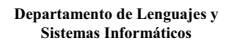
Para recargar la tarjeta el usuario marca la cantidad deseada e introduce el dinero.

Realice un Diagrama de Transición de Estados con el cual se pueda comprobar, junto con el cliente, que hemos comprendido el funcionamiento de la aplicación.

Indique qué tipos de pruebas utilizaría para asegurar que el sistema funciona correctamente.



Autores: José Félix Estívariz López, Rubén Heradio Gil,
Juan Antonio Mascarell Estruch





UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN
Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN
EXTRANJERO RESERVA



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: septiembre 2003

Hora:

Duración: 2 horas

MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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. Reflexione sobre la figura del analista. ¿Se limita a recoger las necesidades del cliente/usuario, investigar soluciones análogas y deducir unas conclusiones que traduce a un lenguaje comprensible por los informáticos y que representa mediante un modelo o tiene que tomar, en ocasiones, algún tipo de decisión? ¿Se le ocurre de qué tipo serían estas decisiones?
2. Explique brevemente en qué consisten y qué tipos hay de pruebas de caja transparente.

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

3. Se desea construir un sistema informático que automatice la gestión de los empleados, departamentos y proyectos que se realizan en una empresa.

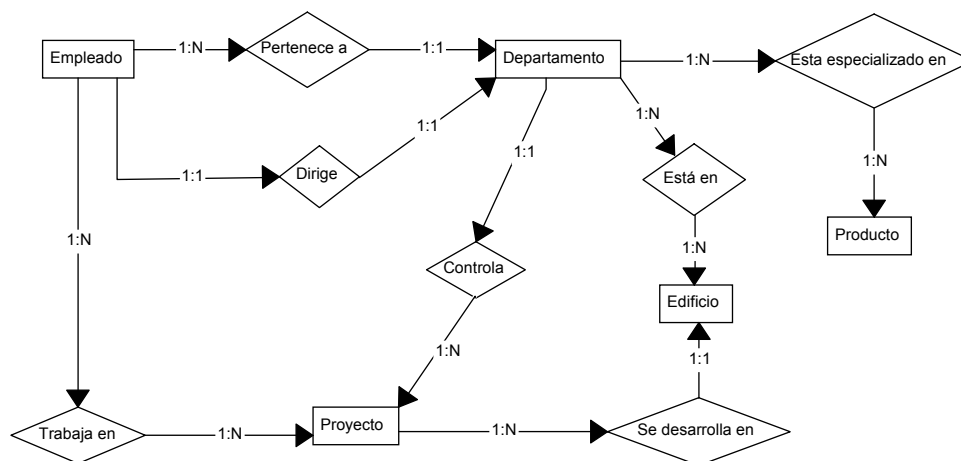
La empresa en cuestión, se organiza en departamentos. Cada departamento dispone de cierto número de empleados y de un director. Los departamentos se especializan en uno o varios productos, aunque puede darse la situación de que más de un departamento esté cualificado para construir un determinado producto. Por otro lado, cada departamento controla cierto número de proyectos. Un empleado está asignado a un sólo departamento, aunque puede trabajar en varios proyectos controlados por otros departamentos. Por último, la empresa dispone de varias sedes. Los departamentos pueden estar repartidos en distintos edificios. Sin embargo, los proyectos se desarrollan exclusivamente en una sede.

Analice el sistema propuesto utilizando un diagrama Entidad-Relación.



Solución propuesta por el equipo docente

Autores: José Félix Estívariz López, Rubén Heradio Gil,
Juan Antonio Mascarell Estruch



Observación: El anterior diagrama se ha elaborado con la herramienta DOME. La siguiente tabla resume la equivalencia entre la notación de dicha herramienta y la propuesta en el libro de la asignatura.

Notación libro	Notación DOME
d	0:1
	1:1
o<	0:N
K	1:N



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN
Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN
EXAMEN EXTRAORDINARIO DE DICIEMBRE



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: 10 de diciembre de 2003 Hora: 11:30 Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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. Defina los siguientes tipos de prueba de software: caja negra, caja transparente, alfa y beta. ¿Las pruebas alfa y beta son de caja negra o transparente?
2. Explique en que consisten las fases de análisis y diseño de un producto software. ¿Cuáles serían las diferencias entre un Diagrama de Transición de Estados utilizado en el análisis respecto a otro usado en el diseño?

SEGUNDA PARTE. PREGUNTA DE APLICACIÓN (5 PUNTOS)

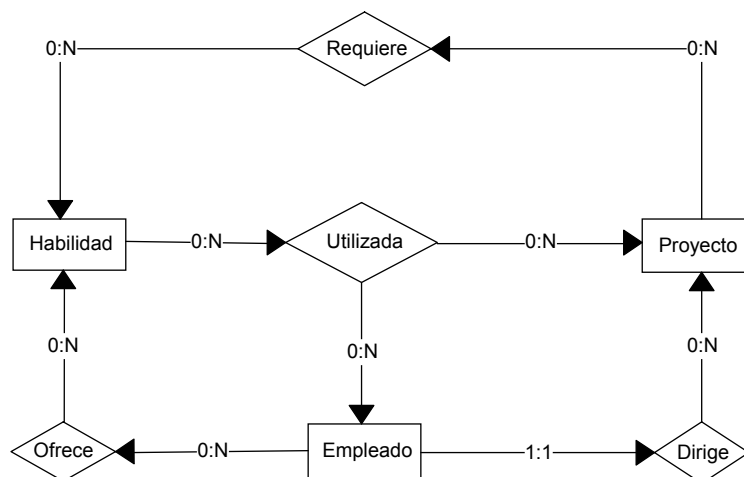
3. Modele mediante un Diagrama Entidad-Relación (DER) la información que se enuncia a continuación sobre los proyectos que se desarrollan en una empresa:

Un empleado puede dirigir varios proyectos, aunque cada proyecto ha de ser dirigido por un solo empleado. La asignación entre proyectos y directores se realiza según las habilidades necesarias para el desarrollo de cada proyecto y las que ofrece cada empleado. El DER deberá reflejar las habilidades que un empleado utiliza en un proyecto concreto.



Solución propuesta por el equipo docente

Autores: José Félix Estívariz López, Rubén Heradio Gil,
Juan Antonio Mascarell Estruch



Observación: El anterior diagrama se ha elaborado con la herramienta DOME. La siguiente tabla resume la equivalencia entre la notación de dicha herramienta y la propuesta en el libro de la asignatura.

Notación libro	Notación DOME
⌞	0:1
	1:1
⌞	0:N
⌞	1:N

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN
NACIONAL, 1ª SEMANA



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: 26 de mayo de 2004

Hora: 11:30

Duración: 2 horas

MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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. Defina y distinga la 'validación' y la 'verificación'. ¿En qué fase del ciclo de vida de cascada se realiza cada una?

Páginas 15 y 16 del libro.

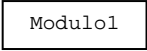
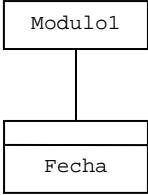
2. ¿Qué tres objetivos fundamentales o cualidades mínimas es deseable alcanzar al hacer la descomposición modular de un sistema? Explique cada uno de ellos y, en cada caso, cómo se pueden medir o qué factores intervienen.

Reflexión sobre el epígrafe 4.1, páginas 150-160 del libro.

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

3. Ejercicio de diseño

Dentro de un sistema informático se emplea el módulo Modulo1. Para su desarrollo se plantean dos diseños alternativos:

	Diseño 1	Diseño 2
		
Código aproximado en Modula-2	<pre>1MODULE Modulo1; 2FROM InOut 3 IMPORT WriteCard, WriteString; 4... 5VAR 6 fecha1, fecha2: 7 ARRAY [1..6] OF TipoDigito; 8 ... 9BEGIN 10 ... 11 (* fecha1 := 30-04-1974 *) 12 fecha1[1] := 3; 13 fecha1[2] := 0; 14 fecha1[3] := 0; 15 fecha1[4] := 4; 16 fecha1[5] := 7; 17 fecha1[6] := 4; 18 (* Imprimir anno *) 19 WriteCard(fecha1[5], 1); 20 WriteCard(fecha1[6], 1); 21 ... 22 (* fecha2 := 10-01-1998 *) 23 fecha2[1] := 1; 24 fecha2[2] := 0; 25 fecha2[3] := 0; 26 fecha2[4] := 1; 27 fecha2[5] := 9; 28 fecha2[6] := 8; 29 ... 30END Modulo1.</pre>	<div><pre>1MODULE Modulo1; 2IMPORT Fecha; 3... 4VAR 5 fecha1, fecha2: Fecha.Tipo; 6 ... 7BEGIN 8 ... 9 Fecha.Crear(fecha1, 30, 4, 1974); 10 Fecha.ImprimirAnno(fecha1); 11 ... 12 Fecha.Crear(fecha2, 10, 1, 1998); 13 ... 14END Modulo1.</pre></div> <div><pre>1DEFINITION MODULE Fecha; 2TYPE Tipo; 3PROCEDURE Crear(VAR fecha: Tipo); 4PROCEDURE ImprimirAnno(fecha: Tipo); 5... 6END Fecha.</pre></div>

Compare los dos diseños analizando cómo aplican los conceptos de Abstracción, Modularidad y Ocultación.

Inicialmente, cuando se planteó el sistema informático, se consideró que para el almacenamiento de los años bastaba con dos dígitos. Sin embargo, con la llegada el nuevo milenio se descubrió que eran necesarios cuatro dígitos. Razone como afectaría este cambio a cada uno de los diseños.

SOLUCIÓN

1. Abstracción

- El primer diseño no utiliza abstracciones de ningún tipo.
- El diseño 2 utiliza el Tipo Abstracto de Datos¹ `Fecha`.

Como resultado, se puede observar que el código asociado al diseño 1 será muy redundante (líneas 12-17 \approx líneas 23-28). La redundancia induce a errores e inconsistencias.

2. Modularidad

El diseño 1 es monolítico, mientras que el diseño 2 es modular. Por ello, el diseño 2 dispone de las siguientes ventajas sobre el diseño 1:

- Permite dividir la implementación entre varias personas (un programador puede codificar `Modulo1` y otro, `Fecha`)
- La implementación asociada al diseño 2 es clara y concisa.
- Los costes asociados al desarrollo, la depuración, la documentación y el mantenimiento del diseño 2 son menores que los del diseño 1.
- El diseño 2 permite reutilizar el concepto de fecha en otros proyectos.

3. Ocultación

En el diseño 1 las “interioridades de las fechas están al descubierto”, es decir, no existe ocultación del concepto de fecha. Este hecho, conlleva dos grandes problemas:

- Si se produce un error en el uso de una fecha, su detección será ardua, ya que a este concepto se accede directamente desde muchos puntos del programa.
- La modificación de la representación del concepto fecha se propagará a muchas partes del código del `Modulo1`. Si tal y como se propone en el enunciado, se decide extender la representación de los años de 2 dígitos a 4, será necesario modificar las líneas 7, 12-28.

En el diseño 2, el concepto de fecha se encapsula y oculta a través de un Tipo Abstracto de Datos. Las consecuencias benignas de esta estrategia son:

- Si se produce un error asociado a una fecha, la búsqueda se limitará al módulo de implementación de `Fecha`.
- Gracias a la ocultación del concepto fecha, la ampliación del número de dígitos de los años implica cambios exclusivamente en el código del módulo de implementación de `Fecha`.

¹Concretamente y según la nomenclatura estudiada en la asignatura Programación 1, el *Tipo Opaco de Datos* `Fecha`.

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

NACIONAL 2ª SEMANA



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: 9 de junio de 2004

Hora: 11:30

Duración: 2 horas

MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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. Supóngase que una organización decide afrontar, por primera vez, un proyecto software en el que no tiene experiencia y que se sitúa en un ámbito desconocido para ella. Desde el punto de vista del ciclo de vida, indique qué alternativas tiene esta organización para culminar el proyecto con éxito y explique cómo pueden, dichas alternativas, mitigar los problemas potenciales con los que la organización se puede encontrar durante el desarrollo.

Reflexión sobre 'Modelos del proceso de desarrollo (Ciclos de Vida)'. En concreto, puntos 1.5 y 1.6 del libro.

2. Defina qué es un TAD (Tipo Abstracto de Datos) y qué repercusiones tiene su uso en el diseño de software.

Tradicionalmente, en la programación imperativa se optaba por separar los programas en dos partes: la de proceso y la de datos (ejemplos de lenguajes imperativos son FORTRAN, COBOL, PASCAL, BASIC...). La parte de proceso accedía y operaba directamente sobre los datos.

Esta separación produce una independencia funcional muy baja. Un ejemplo que ilustra esto es el "efecto 2000", donde la simple adición de dígitos al formato de las fechas supuso realizar muchas modificaciones en la parte de proceso.

Para solventar estos problemas surgió el concepto de Tipo Abstracto de Datos (TAD), que agrupa en una sola entidad la representación de los datos y la parte de proceso que los manipula. Los TADs ocultan la representación de los datos, que sólo es accesible desde las operaciones. De esta forma, un cambio en la representación de los datos de un TAD no se propaga a todo el programa, sino solamente a las operaciones del TAD.

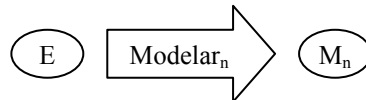
Tradicionalmente, las estructuras de datos se venían definiendo por su representación. Sin embargo, el paso clave hacia la abstracción de los datos es invertir este punto de vista: olvidar por el momento la representación y considerar que las operaciones en sí mismas definen la estructura de datos.

NOTA: un error grave cometido por muchos alumnos consiste en diseñar una operación fuera del TAD que le corresponde. Por ejemplo, sería erróneo el utilizar un TAD llamado “Rotaciones” y una abstracción funcional, separada del TAD, llamada “ProducirRotaciones”.

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

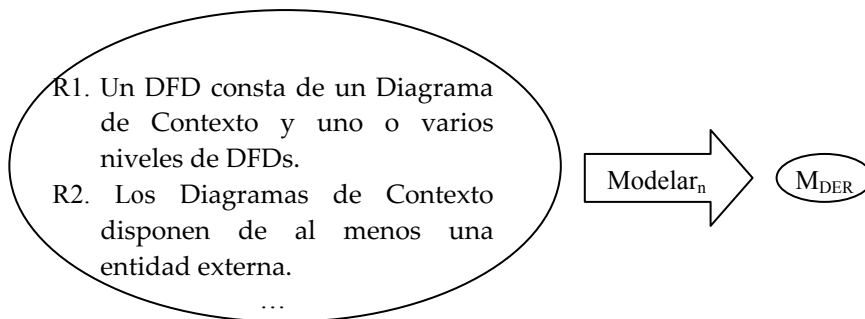
3. Metamodelar DFD con DER

Modelar una especificación E con una notación n , significa obtener un modelo M_n que represente adecuadamente a E .



En el caso del metamodelado, la especificación que se utiliza como entrada es también la correspondiente a una notación de modelado.

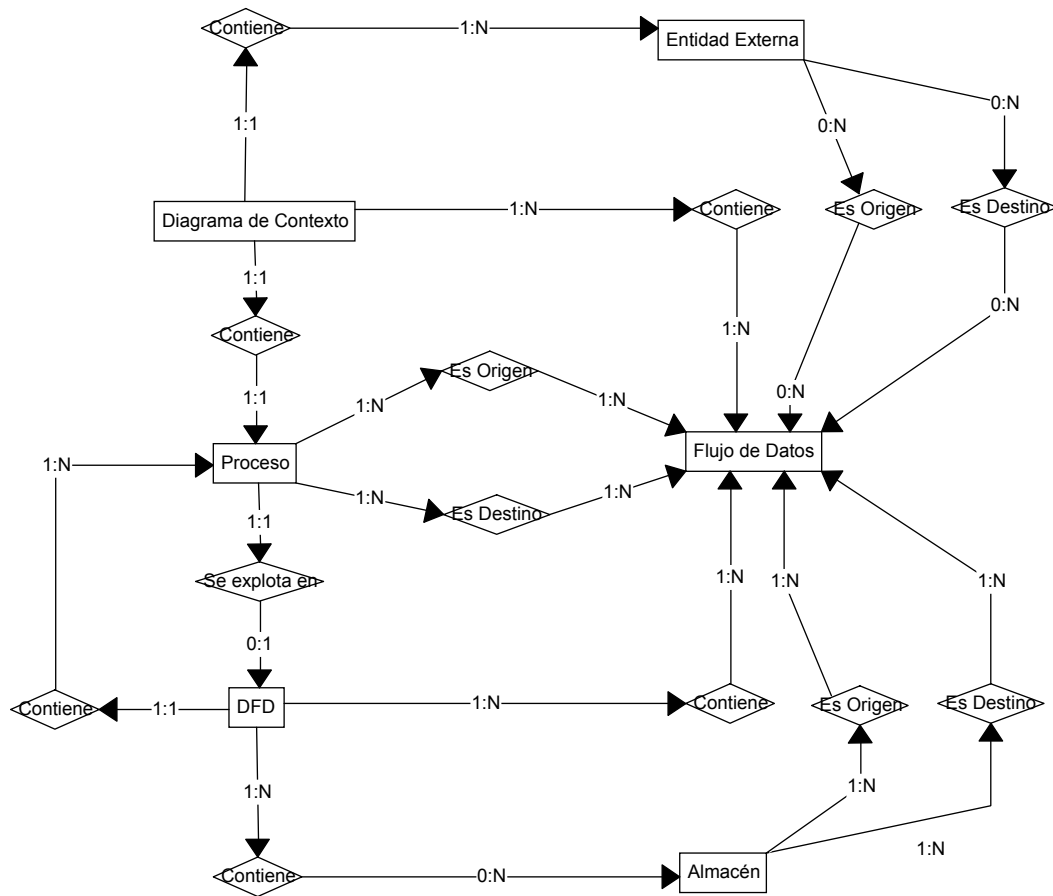
Metamodele con un DER (Diagrama Entidad Relación) la notación DFD (Diagrama de Flujo de Datos), es decir, para la siguiente figura, desarrolle M_{DER} .



AYUDA: Se sugiere que se haga lo siguiente:

1. Escriba una lista con las especificaciones funcionales que reflejen cómo se construye un DFD.
2. Con la lista anterior, construya el modelo utilizando, para ello, la notación Entidad - Relación.

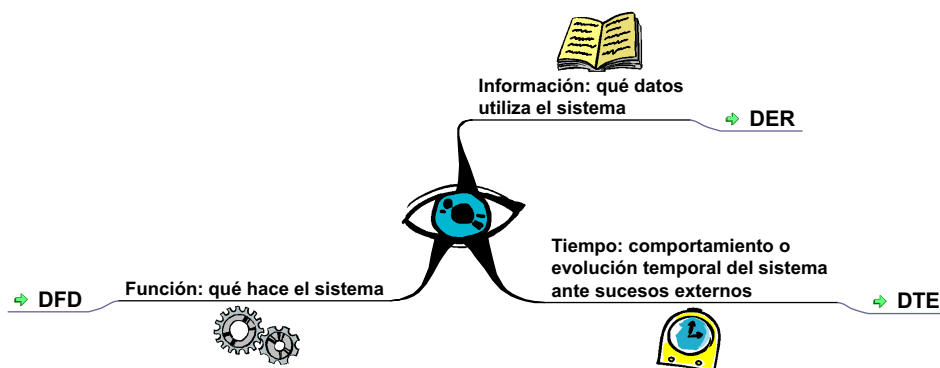
SOLUCIÓN



! Observación

Durante la corrección, hemos detectado que algunos alumnos han interpretado incorrectamente el enunciado y han tratado de buscar una equivalencia entre los DFDs y los DERs. Por ejemplo, los flujos de datos^{DFD} se corresponden con relaciones^{DER}, los almacenes de datos^{DFD} y las entidades externas^{DFD} con entidades^{DER}...

Esto supone un grave error, ya que los DFDs, DERs y DTEs representan distintos puntos de vista (prácticamente ortogonales) de un sistema software, y por tanto, es absurdo buscar su equivalencia.



INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

ORIGINAL EXTRANJERO



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: 9 de junio de 2004

Hora: 11:30

Duración: 2 horas

MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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. Defina el concepto de 'configuración software'. Explique qué es y cómo se utiliza la 'línea base' para gestionar los cambios en la gestión de configuración del software.

Contenido del epígrafe 1.9.5 Gestión de configuración, páginas 30 a 32 del libro.

2. Defina qué es un TAD (Tipo Abstracto de Datos) y qué repercusiones tiene su uso en el diseño de software.

Tradicionalmente, en la programación imperativa se optaba por separar los programas en dos partes: la de proceso y la de datos (ejemplos de lenguajes imperativos son FORTRAN, COBOL, PASCAL, BASIC...). La parte de proceso accedía y operaba directamente sobre los datos.

Esta separación produce una independencia funcional muy baja. Un ejemplo que ilustra esto es el "efecto 2000", donde la simple adición de dígitos al formato de las fechas supuso realizar muchas modificaciones en la parte de proceso.

Para solventar estos problemas surgió el concepto de Tipo Abstracto de Datos (TAD), que agrupa en una sola entidad la representación de los datos y la parte de proceso que los manipula. Los TADs ocultan la representación de los datos, que sólo es accesible desde las operaciones. De esta forma, un cambio en la representación de los datos de un TAD no se propaga a todo el programa, sino solamente a las operaciones del TAD.

Tradicionalmente, las estructuras de datos se venían definiendo por su representación. Sin embargo, el paso clave hacia la abstracción de los datos es invertir este punto de vista: olvidar por el momento la representación y considerar que las operaciones en sí mismas definen la estructura de datos.

NOTA: un error grave cometido por muchos alumnos consiste en diseñar una operación fuera del TAD que le corresponde. Por ejemplo, sería erróneo el utilizar un TAD llamado "Rotaciones" y una abstracción funcional, separada del TAD, llamada "ProducirRotaciones".

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

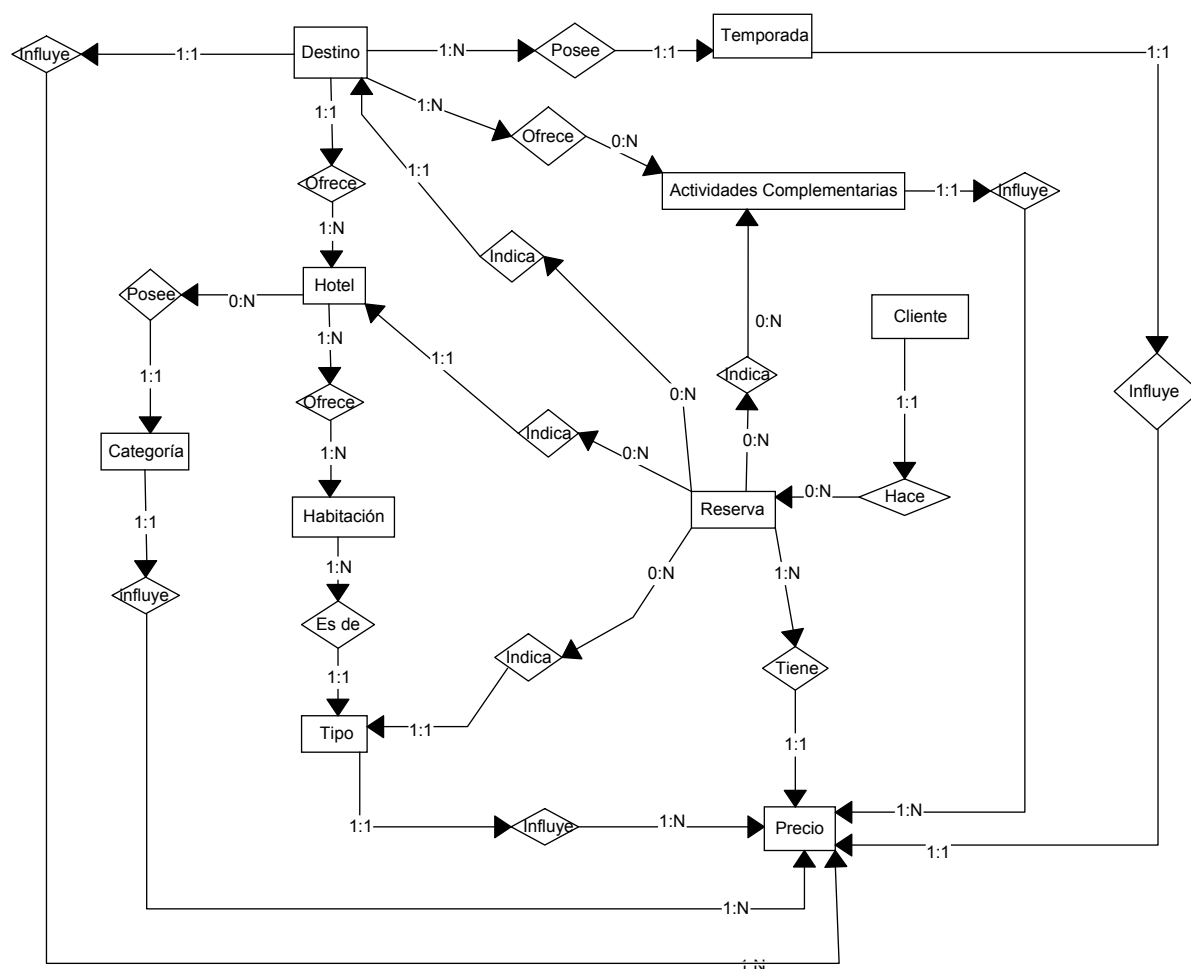
3. Un cliente nos propone desarrollar una aplicación para gestionar algunos aspectos de una agencia de viajes y nos ofrece esta descripción:

“La agencia de viajes ofrece a sus clientes destinos tanto nacionales como en el extranjero. Para cada destino, la agencia ofrece alojamiento en varios hoteles de distinta categoría. Así mismo, se ofrece, para cada destino, una serie de actividades complementarias (excursiones, paseos en barca, etc...).





Si el cliente está de acuerdo, hará una reserva de viaje en la que figurarán los datos del cliente, el destino, tipo de hotel y tipo de habitación (doble, individual, suite, etc.). El precio del viaje, además de las distintas opciones elegidas, depende de la temporada del destino: alta, baja, etc...”

Analice la descripción anterior, elabore una lista de especificaciones funcionales y construya un modelo mediante un diagrama Entidad-Relación.

SOLUCIÓN



Observación: El anterior diagrama se ha elaborado con la herramienta DOME. La siguiente tabla resume la equivalencia entre la notación de dicha herramienta y la propuesta en el libro de la asignatura.

Notación libro	Notación DOME
	0:1
	1:1
	0:N
	1:N

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CENTROS PENITENCIARIOS



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

FECHA: junio de 2004

Hora:

Duración: 2 horas

MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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. Supóngase que una organización decide afrontar, por primera vez, un proyecto software en el que no tiene experiencia y que se sitúa en un ámbito desconocido para ella. Desde el punto de vista del ciclo de vida, indique qué alternativas tiene esta organización para culminar el proyecto con éxito y explique cómo pueden, dichas alternativas, mitigar los problemas potenciales con los que la organización se puede encontrar durante el desarrollo.

Reflexión sobre 'Modelos del proceso de desarrollo (Ciclos de Vida)'. En concreto, puntos 1.5 y 1.6 del libro.

2. Defina qué es un TAD (Tipo Abstracto de Datos) y qué repercusiones tiene su uso en el diseño de software.

Tradicionalmente, en la programación imperativa se optaba por separar los programas en dos partes: la de proceso y la de datos (ejemplos de lenguajes imperativos son FORTRAN, COBOL, PASCAL, BASIC...). La parte de proceso accedía y operaba directamente sobre los datos.

Esta separación produce una independencia funcional muy baja. Un ejemplo que ilustra esto es el "efecto 2000", donde la simple adición de dígitos al formato de las fechas supuso realizar muchas modificaciones en la parte de proceso.

Para solventar estos problemas surgió el concepto de Tipo Abstracto de Datos (TAD), que agrupa en una sola entidad la representación de los datos y la parte de proceso que los manipula. Los TADs ocultan la representación de los datos, que sólo es accesible desde las operaciones. De esta forma, un cambio en la representación de los datos de un TAD no se propaga a todo el programa, sino solamente a las operaciones del TAD.

Tradicionalmente, las estructuras de datos se venían definiendo por su representación. Sin embargo, el paso clave hacia la abstracción de los datos es invertir este punto de vista: olvidar por el momento la representación y considerar que las operaciones en sí mismas definen la estructura de datos.

NOTA: un error grave cometido por muchos alumnos consiste en diseñar una operación fuera del TAD que le corresponde. Por ejemplo, sería erróneo el utilizar un TAD llamado “Rotaciones” y una abstracción funcional, separada del TAD, llamada “ProducirRotaciones”.

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

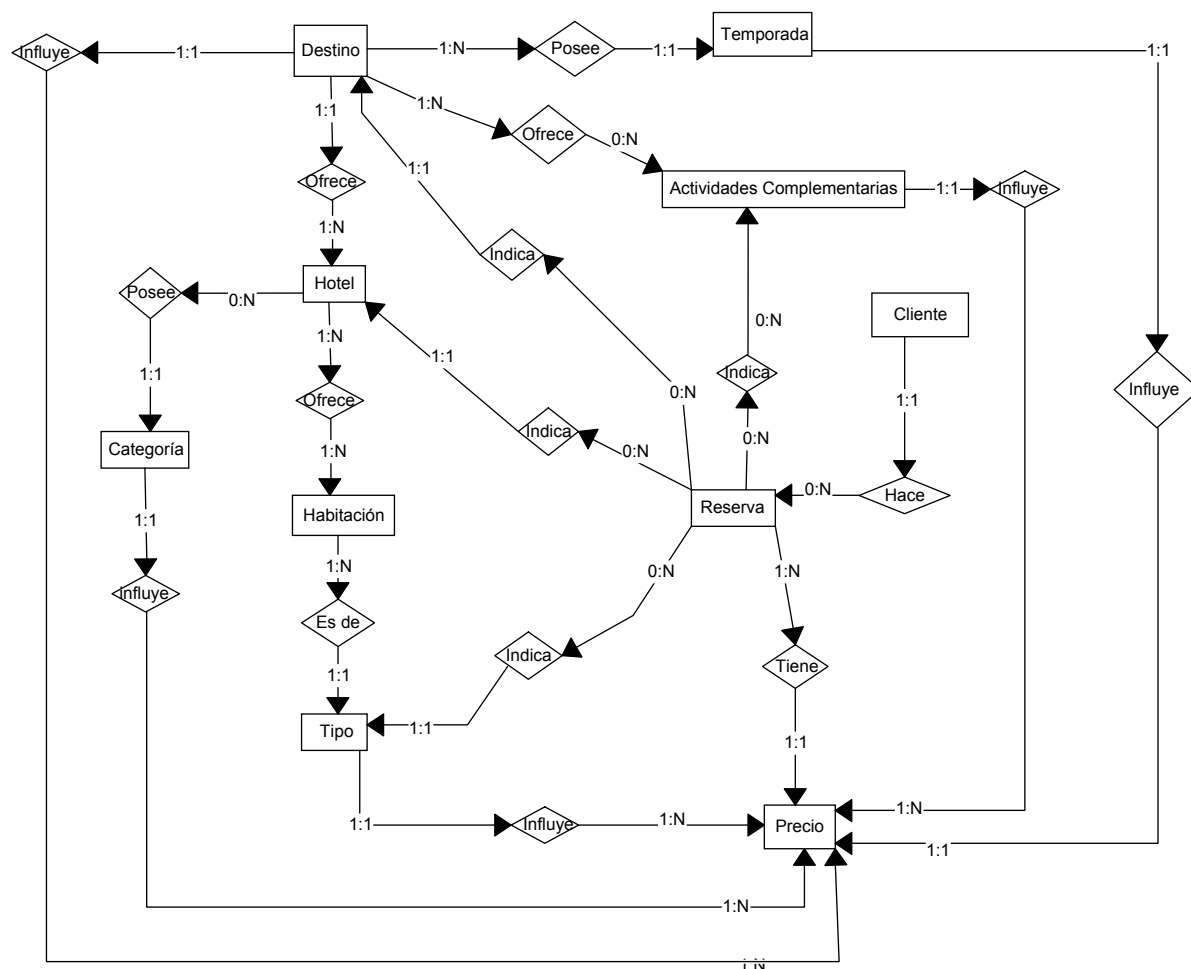
3. Un cliente nos propone desarrollar una aplicación para gestionar algunos aspectos de una agencia de viajes y nos ofrece esta descripción:

“La agencia de viajes ofrece a sus clientes destinos tanto nacionales como en el extranjero. Para cada destino, la agencia ofrece alojamiento en varios hoteles de distinta categoría. Así mismo, se ofrece, para cada destino, una serie de actividades complementarias (excursiones, paseos en barca, etc...).

Si el cliente está de acuerdo, hará una reserva de viaje en la que figurarán los datos del cliente, el destino, tipo de hotel y tipo de habitación (doble, individual, suite, etc.). El precio del viaje, además de las distintas opciones elegidas, depende de la temporada del destino: alta, baja, etc...”

Analice la descripción anterior, elabore una lista de especificaciones funcionales y construya un modelo mediante un diagrama Entidad-Relación.

SOLUCIÓN



Observación: El anterior diagrama se ha elaborado con la herramienta DOME. La siguiente tabla resume la equivalencia entre la notación de dicha herramienta y la propuesta en el libro de la asignatura.

Notación libro	Notación DOME
0	0:1
1	1:1
∞	0:N
K	1:N

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

ORIGINAL NACIONAL



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: **INGENIERÍA DEL SOFTWARE (2º CURSO)**

FECHA: 4 de septiembre de 2004 Hora: 11:30 Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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) (o 4 caras A4, un pliego, un A3 doblado en forma de carpeta, etc.)

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

1. Reflexione y explique cómo aplicaría el método de Abbott para analizar un sistema y para construir un modelo con la notación de los DFD.

SOLUCIÓN

La técnica de Abbott establece un procedimiento sistemático para la obtención de elementos significativos de un sistema a partir de una descripción de un modelo. En el caso del diseño, si lo que estamos buscando son abstracciones, partimos de la caracterización o definición de lo que estamos buscando y del espacio de búsqueda (en el libro, una descripción en lenguaje natural) En el diseño se trata de construir una descripción del funcionamiento de la aplicación. Para que dicha aplicación, y su funcionamiento, cumplan con determinados objetivos, se debe construir la descripción descomponiéndola en módulos. Cuando se utilizan abstracciones para el diseño, la descomposición y la descripción vienen caracterizados por ciertos elementos significativos (las abstracciones). Así, al aplicar Abbott en el diseño mediante abstracciones, si lo que se intenta es identificar la existencia de, por ejemplo, una abstracción funcional (la cual no es sino un conjunto de acciones y operaciones que tienen un objetivo concreto); se buscará en la descripción en lenguaje natural el elemento del código lingüístico indica una acción u operación: los verbos. La técnica de Abbott dice: busquemos verbos porque, seguramente, nos ayudarán a localizar las abstracciones funcionales de nuestro sistema.

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

El método de Abbott, en realidad, es una forma de análisis. Si se aplica el método descrito anteriormente a la construcción de DFDs, se comprueba su utilidad: ¿qué se busca? ¿procesos? Y ¿qué son los procesos? Actividades y tareas que tendrá que realizar nuestra aplicación. ¿Cuál es el espacio de búsqueda en este caso? La comprensión de lo que tendrá que hacer nuestra aplicación. ¿Cómo representamos en nuestra mente a las actividades y tareas? Mediante verbos o cualquier representación que utilicemos internamente para la idea de acción o tarea. Pues esta es una manera de encontrar los procesos o ‘bolas’ de un DFD. Se haría igual con los flujos de datos o las entidades externas, etc., sólo que, ahora, buscaremos descripciones de elementos que interactúan con el sistema, sustantivos o adjetivos, etc., que se van a procesar, transformar o almacenar, en el sistema que estamos construyendo.

2. Defina y estructure una clasificación de las técnicas generales para el diseño procedimental del software. Utilice un esquema arbóreo; identifique cada técnica y explique sus características principales. (Hasta aquí la pregunta. La mayor parte de las técnicas de diseño provienen de la programación, lo cual se explica -de forma más o menos implícita- en el desarrollo del libro. Por ello, se valorará positivamente que el estudio anterior se compare y diferencie respecto a otro análogo pero para las técnicas de codificación.)

SOLUCIÓN: (Páginas 160 a 187 del libro. Epígrafes 4.2 a 4.4)

I. Diseño procedimental

1 Diseño funcional descendente

1.1 Técnica del refinamiento progresivo

Aplicación, a la fase de diseño, del concepto de refinamientos sucesivos [Wirth]. Consiste en plantear la aplicación como única operación global, e irla descomponiendo en operaciones más sencillas, detalladas y específicas. En cada nivel de refinamiento, las operaciones identificadas se asignan a módulos separados.

La construcción de programas mediante refinamientos sucesivos proviene de la metodología de programación estructurada [Dijkstra]. Esta metodología de programación consiste en la utilización sólo de estructuras de control claras y sencillas, con esquemas de ejecución con un único punto de inicio y un único punto final.

1.2 Técnica de programación estructurada de Jackson

Aplicación, a la fase de diseño, de la metodología de programación estructurada [Dijkstra]. Esta metodología de programación consiste en la utilización sólo de estructuras de control claras y sencillas, con esquemas de ejecución con un único punto de inicio y un único punto final. Dichas estructuras son la **secuencia**, la **selección** y la **iteración**.

La técnica de Jackson consiste en construir las estructuras del programa de forma similar a las estructuras de los datos de entrada-salida que se manejan.

1.3 Técnica de diseño estructurado [Yourdon]

Técnica de diseño complementaria del ‘análisis estructurado’ (utilización de DFD). Consiste en:

1. Utilización de DFD para la descripción del funcionamiento de la aplicación.
2. Con los primeros niveles de la descomposición en DFD, se construye un único diagrama en el que se incluyen los procesos implicados y se prescinde de los elementos de

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

almacenamiento. Es este último diagrama, se realiza un análisis para identificar o bien un único flujo global (flujo de transformación) o bien uno o varios puntos en los que el flujo global se bifurca (flujo de transacción).

3. Según el patrón identificado en el análisis anterior, se construye el diagrama de diseño mediante la notación de los diagramas de estructura. Para ello, a partir de los DFD iniciales, se asignan procesos o grupos de procesos a los módulos de diseño y se establece una jerarquía o estructura de control en función del mencionado patrón identificado en el punto anterior (transformación o transacción).

2 Diseño con abstracciones

2.1 *Descomposición modular con abstracciones*

Consiste en dedicar módulos separados a la realización de cada tipo abstracto de datos y cada función importante. Se emplea la notación de los diagramas de bloques jerarquizados; en los que se representan las relaciones de uso y la jerarquía se establece de arriba hacia abajo ('los de arriba usan a los de abajo').

Se puede aplicar de forma descendente (ampliación del refinamiento progresivo con las abstracciones) o ascendente (ampliación de primitivas hacia abstracciones de nivel superior).

2.2 *Método de Abbott*

Metodología para identificar los elementos abstractos que formarán parte del diseño. Las abstracciones obtenidas se complementan y se organizan en un diagrama de diseño. El método se suele aplicar a las descripciones del análisis (formales o informales).

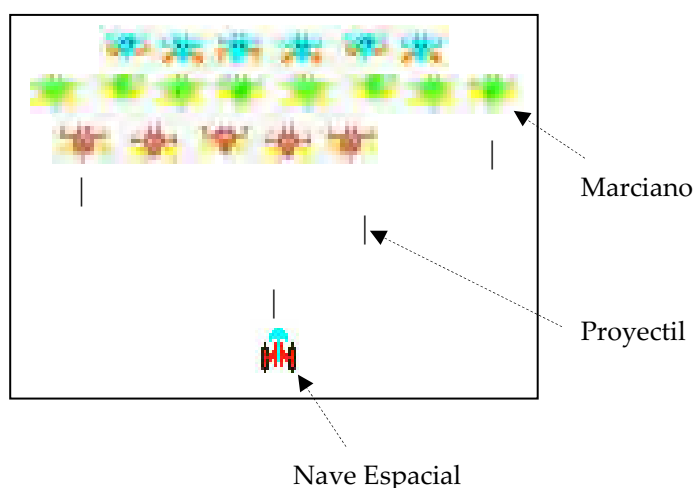
3 Diseño orientado a objetos

En esencia es similar al diseño con abstracciones pero, además de la relaciones de uso y agregación, hay que considerar la herencia y el polimorfismo. En la descomposición modular del sistema, cada módulo contiene la descripción de una clase o de varias clases de objetos relacionadas entre sí. Se suele utilizar un diagrama de estructura para describir la arquitectura del sistema y las relaciones de uso y diagramas ampliados de las clases y objetos en las que se representan las distintas relaciones entre los datos (herencia, polimorfismo, etc.).

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

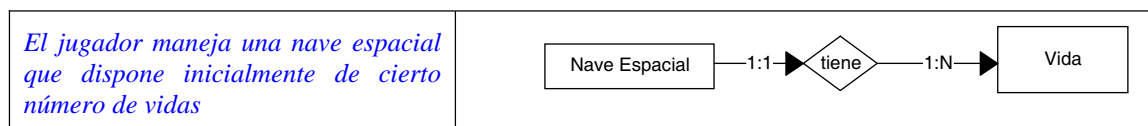
4 Analice mediante un DER (Diagrama Entidad-Relación) la siguiente especificación correspondiente a un videojuego tipo “marcianitos” como el de la figura.

El jugador maneja una nave espacial que dispone inicialmente de cierto número de vidas. La nave debe enfrentarse a distintos tipos de “marcianos” enemigos. Tanto la nave como los marcianos lanzan proyectiles. Cuando la nave recibe un impacto, pierde una de sus vidas. Cuando el n° de vidas de la nave es cero, finaliza la partida. Cada tipo de marciano es capaz de resistir cierto n° de impactos. Algunos marcianos, al ser destruidos desprenden un “bonus”; si la nave captura un bonus, incrementa en uno su número de vidas.



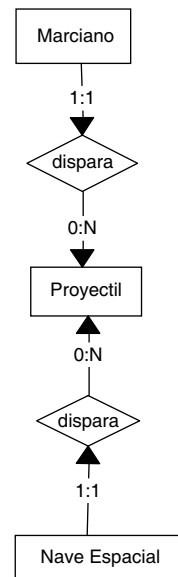
Solución

La construcción del DER solicitado puede abordarse gradualmente. Del texto de la especificación han de extraerse las entidades, las relaciones y la cardinalidad de estas últimas.

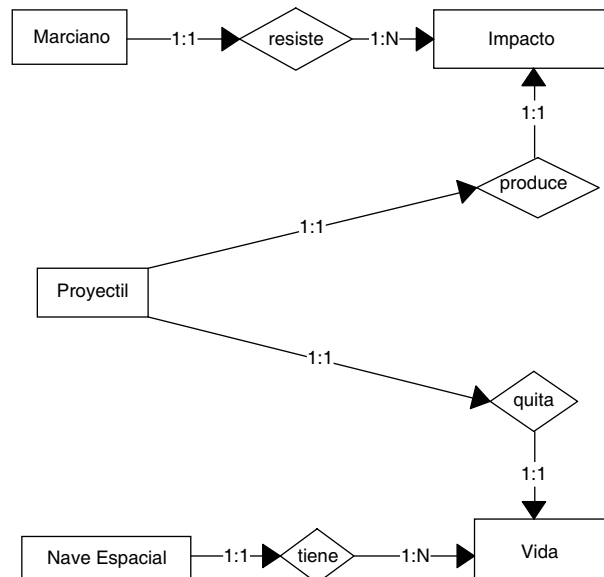


Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

La nave debe enfrentarse a distintos tipos de “marcianos” enemigos. Tanto la nave como los marcianos lanzan proyectiles.



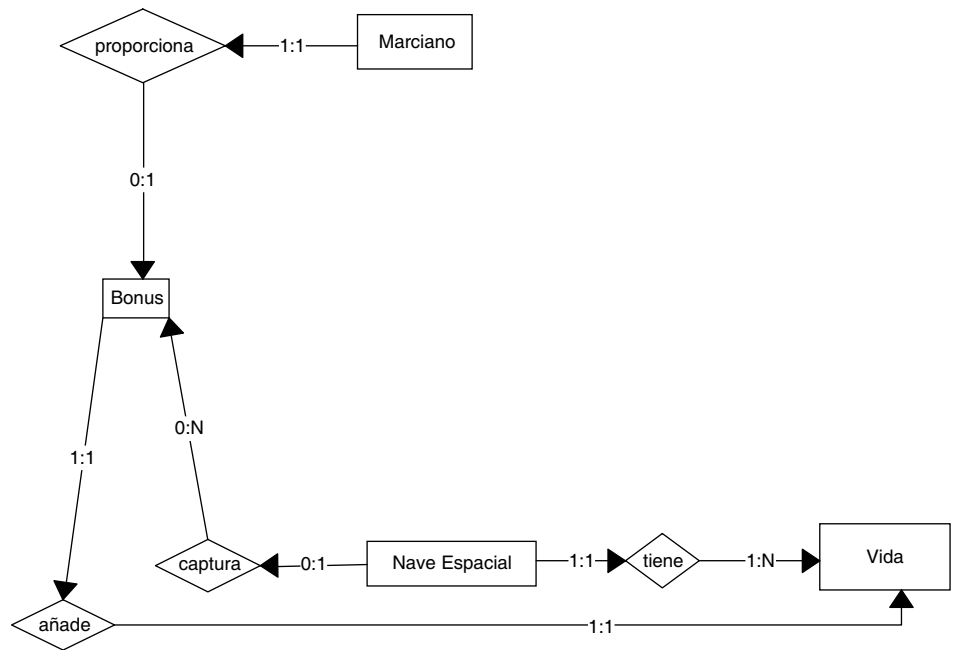
Cuando la nave recibe un impacto, pierde una de sus vidas. Cuando el n° de vidas de la nave es cero, finaliza la partida. Cada tipo de marciano es capaz de resistir cierto n° de impactos.



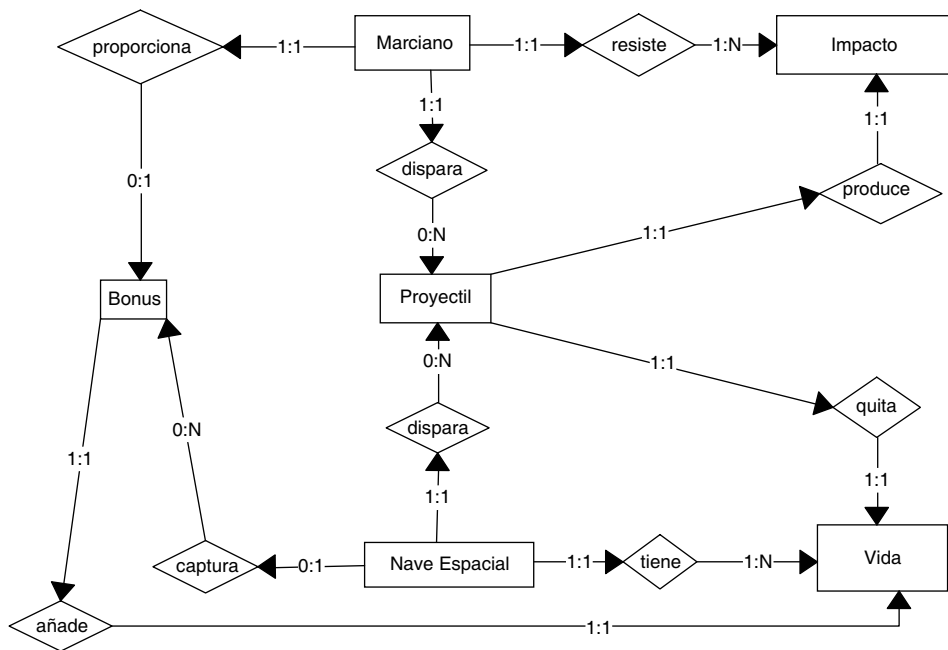
Observación: los DER son una notación estática útil para modelar los aspectos estructurales de una especificación. Los aspectos temporales se pueden modelar con una notación dinámica como los DTE (Diagramas de Transición de Estados). Por eso, la frase del enunciado *Cuando el n° de vidas de la nave es cero, finaliza la partida* no se modela en el DER correspondiente.

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

Algunos marcianos, al ser destruidos desprenden un "bonus"; si la nave captura un bonus, incrementa en uno su número de vidas.



El resultado final del análisis es el siguiente DER:



Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN
NACIONAL RESERVA



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: **INGENIERÍA DEL SOFTWARE (2º CURSO)**

FECHA: 8 de septiembre de 2004 Hora: 9:00 Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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) (o 4 caras A4, un pliego, un A3 doblado en forma de carpeta, etc.)

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

1. Explique las propiedades que debe tener el modelo de un sistema software para lograr una especificación correcta.

SOLUCIÓN

Punto 2.2.1 del libro de texto. Páginas 43 a 46.

2. Explique por qué es, la descomposición modular, uno de los objetivos principales del diseño del software.

SOLUCIÓN

Punto 4.1 del libro de texto. Páginas 148 a 160 (resumen).

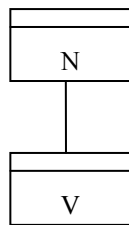
Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

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

3. Se dispone de un Tipo Abstracto de Datos (TAD) V , con las operaciones $op1$, $op2$ y $op3$. Razone cómo se podría derivar un nuevo tipo TAD N (utilizando exclusivamente elementos propios de un Diagrama de Abstracciones) que tuviera las operaciones $op1$, $op2$, $op3$ y $op4$, donde:
- $op1^N$ ($op1$ del TAD N) sería idéntica a $op1^V$ ($op1$ del TAD V)
 - $op2^N$ tendría un comportamiento distinto a $op2^V$
 - $op3^N$ sería idéntica a $op3^V$
 - $op4^N$ sería totalmente nueva
- ¿Cuál sería la solución si V y N fueran Clases de Objetos y se aplicarían Herencia y Polimorfismo?

SOLUCIÓN

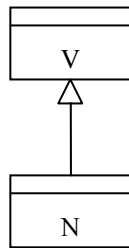
a) N y V como TADs



Lista de operaciones de N :

Operación	Pseudocódigo
$op1^N$	llamar a $op1^V$
$op2^N$	<<nuevo código>>
$op3^N$	llamar a $op3^V$
$op4^N$	<<nuevo código>>

b) N y V como Clases de Objetos



Lista de operaciones de N :

Operación	Pseudocódigo
$op2^N$	<<nuevo código>>
$op4^N$	<<nuevo código>>

Como N hereda de V , N adquiere automáticamente las operaciones de V . Por lo tanto, no sería necesario reescribir las operaciones $op1$ y $op3$ en N . En cambio, sí habría que añadir $op2^N$, que sería invocada adecuadamente gracias al polimorfismo de anulación. Finalmente, también habría que añadir a N la nueva operación $op4^N$.

INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

ORIGINAL EXTRANJERO



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: **INGENIERÍA DEL SOFTWARE (2º CURSO)**

FECHA: 4 de septiembre de 2004 Hora: Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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) (o 4 caras A4, un pliego, un A3 doblado en forma de carpeta, etc.)

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

1. Explique las ventajas del uso de prototipos, en sus distintas variantes, durante el ciclo de vida.

SOLUCIÓN

Punto 1.5 del libro de texto. Páginas 16, 17 y, parcialmente, las siguientes de este punto.

2. Defina y estructure una clasificación de las técnicas generales para el diseño procedimental del software. Utilice un esquema arbóreo; identifique cada técnica y explique sus características principales. (Hasta aquí la pregunta. La mayor parte de las técnicas de diseño provienen de la programación, lo cual se explica -de forma más o menos implícita- en el desarrollo del libro. Por ello, se valorará positivamente que el estudio anterior se compare y diferencie respecto a otro análogo pero para las técnicas de codificación.)

SOLUCIÓN

Páginas 160 a 187 del libro. Epígrafes 4.2 a 4.4

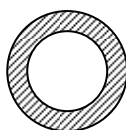
Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

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

3. Se desea construir un programa informático para el dibujo de figuras geométricas planas. La siguiente tabla resume las figuras que manejará el programa y los elementos necesarios para la determinación de cada una de ellos.

Figura	Datos que la determinan
Punto	Dos coordenadas cartesianas
Recta	Dos puntos
Rectángulo	Dos puntos (que determinan su diagonal)
Círculo	Un punto (su centro) y su radio (un número real que representa una longitud)

- a) Realice un diseño mediante un Diagrama Orientado a Objetos que represente las figuras anteriores y sus relaciones.
- b) Se desea añadir al programa anterior el manejo de la figura “corona circular” (ver figura). Indique cómo puede obtenerse esta nueva figura aprovechando el desarrollo anterior mediante:
- Herencia
 - Composición



Corona Circular

SOLUCIÓN

Parte a)

De nuestro enunciado se desprende rápidamente que vamos a tener las siguientes clases:

CLASE Punto

ATRIBUTOS

coordenadaX : Tipo numérico

coordenadaY : Tipo numérico

OPERACIONES

....

FIN-CLASE

CLASE Recta

ATRIBUTOS

punto1 : Tipo Punto

punto2 : Tipo Punto

OPERACIONES

...

FIN-CLASE

CLASE Rectangulo

ATRIBUTOS

derechaSup : Tipo Punto

izquierdaInf : Tipo Punto

OPERACIONES

....

FIN-CLASE

CLASE Circulo

ATRIBUTOS

radio : Tipo numérico

centro : Tipo Punto

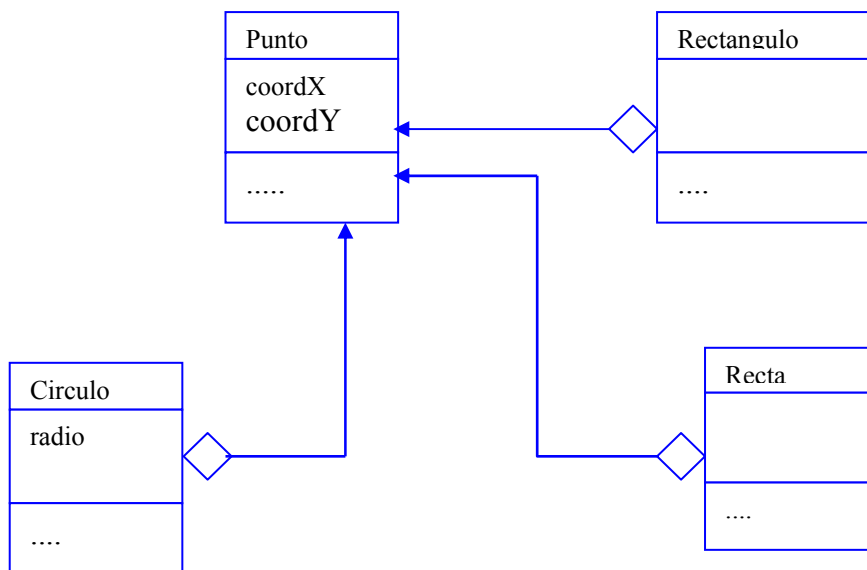
OPERACIONES

....

FIN-CLASE

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

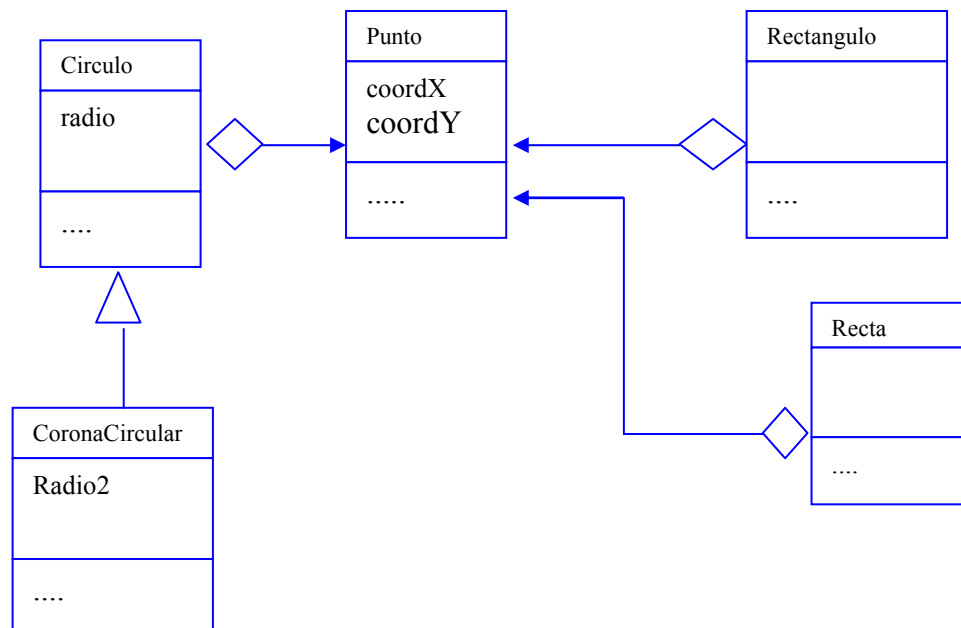
Y el diagrama de clases que representa esto sería:



Parte b)

Solución I.

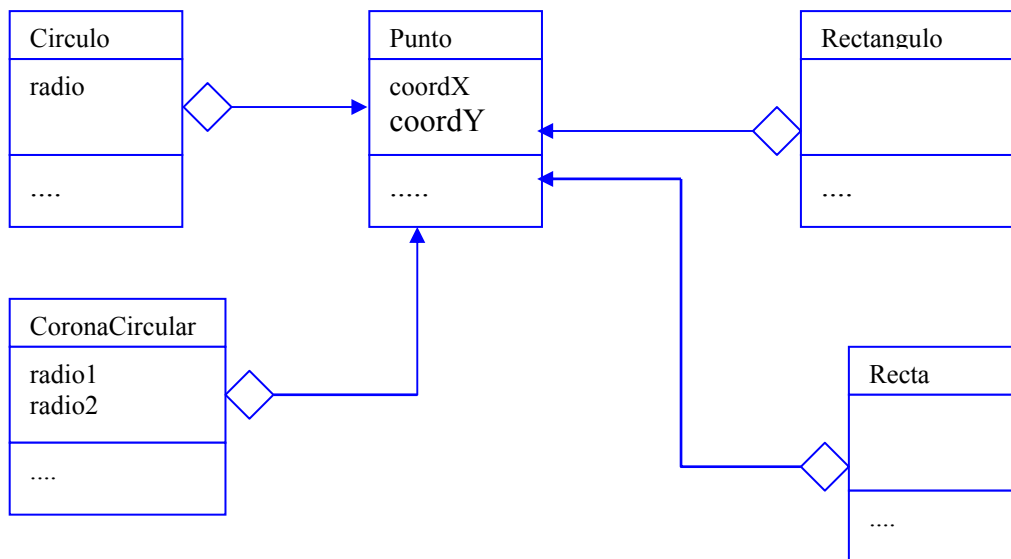
Podemos pensar que una *corona circular* **ES UN** *circulo* con dos radios, esto es, la nueva clase **CoronaCircular** puede aprovechar todas las características de la clase **Circulo**, añadiendo a su vez un nuevo radio, quedando perfectamente determinada toas las características de la corona circular. Todas las propiedades de la clase círculo las obtenemos por tanto mediante herencia, pero además, añadiremos un atributo tipo numérico que representa el segundo radio. El siguiente diagrama representa esta situación:



Solución II.

Por otra parte, podemos 'visualizar' una corona circular como la diferencia de dos círculos. Por tanto, la clase CoronaCircular2 será una clase que contenga dos atributos (en su interior) de tipo Circulo, esto es, la CoronaCircular2 está compuesta de dos círculos.

La figura siguiente refleja la solución II:



INGENIERÍA TÉCNICA en INFORMÁTICA de SISTEMAS y de GESTIÓN



UNIVERSIDAD NACIONAL DE
EDUCACIÓN A DISTANCIA

Plan de estudios en extinción
CÓDIGO CARRERA: 40=SISTEMAS y 41=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

Plan de estudios NUEVO
CÓDIGO CARRERA: 53=SISTEMAS y 54=GESTIÓN
CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

RESERVA EXTRANJERO



Departamento de Lenguajes
y Sistemas Informáticos

ASIGNATURA: **INGENIERÍA DEL SOFTWARE (2º CURSO)**

FECHA: 8 de septiembre de 2004 Hora: Duración: 2 horas MATERIAL: NINGUNO

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En la primera parte (las preguntas teóricas que se valoran con 2'5 puntos cada una) 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) (o 4 caras A4, un pliego, un A3 doblado en forma de carpeta, etc.)

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

1. Enuncie, defina y explique los tres tipos de mantenimiento que se estudian en esta asignatura.

SOLUCIÓN

Punto 1.8.1 del libro de texto. Página 24.

2. Reflexione y explique cómo aplicaría el método de Abbott para analizar un sistema y para construir un modelo con la notación de los DFD.

SOLUCIÓN

La técnica de Abbott establece un procedimiento sistemático para la obtención de elementos significativos de un sistema a partir de una descripción de un modelo. En el caso del diseño, si lo que estamos buscando son abstracciones, partimos de la caracterización o definición de lo que estamos buscando y del espacio de búsqueda (en el libro, una descripción en lenguaje natural) En el diseño se trata de construir una descripción del funcionamiento de la aplicación. Para que dicha aplicación, y su funcionamiento, cumplan con determinados objetivos, se debe construir la descripción descomponiéndola en módulos. Cuando se utilizan abstracciones para el diseño, la descomposición y la descripción vienen caracterizados por ciertos elementos significativos (las abstracciones). Así, al aplicar Abbott en el diseño mediante abstracciones, si lo que se intenta es identificar la existencia de, por ejemplo, una abstracción funcional (la cual no es sino un conjunto de acciones y operaciones que tienen un objetivo concreto); se buscará en la descripción en lenguaje natural el elemento del código lingüístico indica una acción u operación: los verbos. La técnica de Abbott dice: busquemos verbos porque, seguramente, nos ayudarán a localizar las abstracciones funcionales de nuestro sistema.

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

El método de Abbott, en realidad, es una forma de análisis. Si se aplica el método descrito anteriormente a la construcción de DFDs, se comprueba su utilidad: ¿qué se busca? ¿procesos? Y ¿qué son los procesos? Actividades y tareas que tendrá que realizar nuestra aplicación. ¿Cuál es el espacio de búsqueda en este caso? La comprensión de lo que tendrá que hacer nuestra aplicación. ¿Cómo representamos en nuestra mente a las actividades y tareas? Mediante verbos o cualquier representación que utilicemos internamente para la idea de acción o tarea. Pues esta es una manera de encontrar los procesos o ‘bolas’ de un DFD. Se haría igual con los flujos de datos o las entidades externas, etc., sólo que, ahora, buscaremos descripciones de elementos que interactúan con el sistema, sustantivos o adjetivos, etc., que se van a procesar, transformar o almacenar, en el sistema que estamos construyendo.

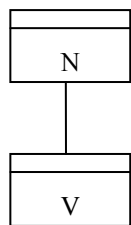
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Se dispone de un Tipo Abstracto de Datos (TAD) V , con las operaciones $op1$, $op2$ y $op3$. Razone cómo se podría derivar un nuevo tipo TAD N (utilizando exclusivamente elementos propios de un Diagrama de Abstracciones) que tuviera las operaciones $op1$, $op2$, $op3$ y $op4$, donde:
- $op1^N$ ($op1$ del TAD N) sería idéntica a $op1^V$ ($op1$ del TAD V)
 - $op2^N$ tendría un comportamiento distinto a $op2^V$
 - $op3^N$ sería idéntica a $op3^V$
 - $op4^N$ sería totalmente nueva

¿Cuál sería la solución si V y N fueran Clases de Objetos y se aplicaran Herencia y Polimorfismo?

SOLUCIÓN

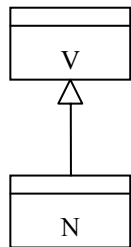
a) N y V como TADs



Lista de operaciones de N :

Operación	Pseudocódigo
$op1^N$	llamar a $op1^V$
$op2^N$	<<nuevo código>>
$op3^N$	llamar a $op3^V$
$op4^N$	<<nuevo código>>

b) N y V como Clases de Objetos



Lista de operaciones de N :

Operación	Pseudocódigo
$op2^N$	<<nuevo código>>
$op4^N$	<<nuevo código>>

Como N hereda de V , N adquiere automáticamente las operaciones de V . Por lo tanto, no sería necesario reescribir las operaciones $op1$ y $op3$ en N . En cambio, sí habría que añadir $op2^N$, que sería invocada adecuadamente gracias al polimorfismo de anulación. Finalmente, también habría que añadir a N la nueva operación $op4^N$.

Entregue la hoja de lectura óptica con sus datos junto con las respuestas de su examen.

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:

Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN

Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: NACIONAL 1ª SEMANA



Departamento de Ingeniería de
Software y Sistemas Informáticos

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En las preguntas teóricas, que se valoran con 2'5 puntos cada una, 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. ¿Qué es la línea base? ¿Para qué sirve? ¿Cómo se usa?

SOLUCIÓN

Epígrafe 1.9.5 “Gestión de la configuración”, ‘*Control de cambios*’, los dos últimos párrafos de la página 31 hasta el final del epígrafe en la página 32.

2. ¿Por qué se busca que la descomposición modular de un diseño cumpla con las cualidades de “Independencia Funcional”, “Comprensibilidad” y “Adaptabilidad”?

SOLUCIÓN

Además de conseguir el objetivo fundamental del diseño, que es la descripción/especificación adecuada (es decir, que permita la implementación en un lenguaje) del funcionamiento de la aplicación; dicho diseño debe permitir un mantenimiento sencillo y un desarrollo que economice recursos. Para ello se hace la ‘descomposición modular’. En las páginas 150 a 160, se encuentra la justificación de en qué medida, las tres cualidades del enunciado, contribuyen a conseguir ese mantenimiento sencillo e implementación con ahorro de recursos.

Entregue la hoja de lectura óptica con sus datos junto con su examen.

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

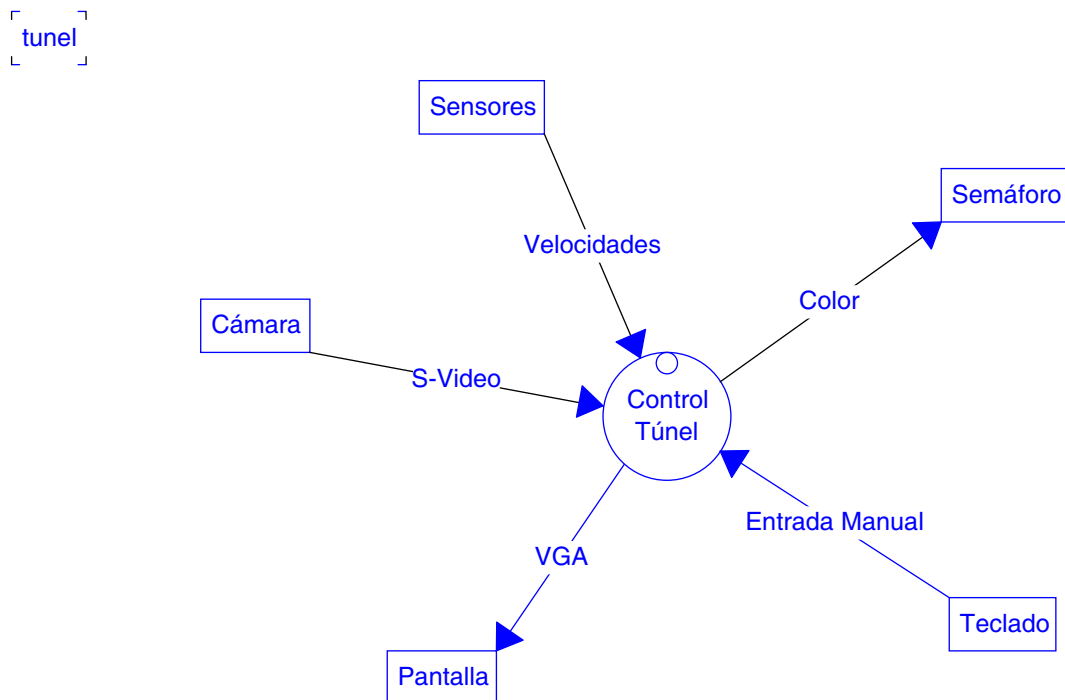
3. Se desea realizar una aplicación informática para evitar congestiones de automóviles en el interior de un túnel. El sistema recibe datos de la velocidad de los automóviles provenientes de un conjunto de sensores distribuidos en el interior del túnel. Los datos de todos los sensores se reciben en paralelo y de forma periódica. El sistema deberá actuar, en función de dichos datos, sobre un semáforo situado a la entrada del túnel, de forma que cuando se detecte un coche circulando a una velocidad inferior a una estipulada, el semáforo se pondrá en rojo y cuando las velocidades detectadas por los sensores sean superiores a otro valor estipulado, se volverá a poner en verde. Existirá un centro de control con una pantalla que muestre el estado del semáforo y las velocidades instantáneas captadas por el conjunto de sensores. El sistema podrá funcionar en modo manual, para lo cual el operario deberá introducir por teclado su identificador y contraseña cada vez que quiera pasar de automático a manual, o viceversa, y/o cambiar el estado del semáforo. Una o varias cámaras de video enviarán imágenes del interior y exterior del túnel, que se mostrarán en la pantalla y servirán para ayudar al operario a tomar decisiones cuando realice el control manual. El sistema almacenará todos los datos sobre velocidades proporcionadas por los sensores, el estado del semáforo, los cambios de estado que se produzcan en él y su origen (automático o manual) y los accesos de los usuarios, junto con la fecha y hora de cada evento.

Se pide: analizar el sistema mediante Diagramas de Flujo de Datos (DFDs), desarrollando los DFDs de contexto, nivel 0 y 1.

SOLUCIÓN

Errata: DFD de contexto = DFDs de nivel 0. Evidentemente, lo que se pedía es ‘**nivel 0 y 1**’ o bien ‘**DFD de contexto y nivel 1**’.

DFD de contexto:



Aplicación informática para evitar congestiones de automóviles en el interior de un túnel. El sistema recibe datos de la velocidad de los automóviles provenientes de un conjunto de sensores distribuidos en el interior del túnel. Los datos de todos los sensores se reciben en paralelo y de forma periódica. El sistema deberá actuar, en función de dichos datos, sobre un semáforo situado a

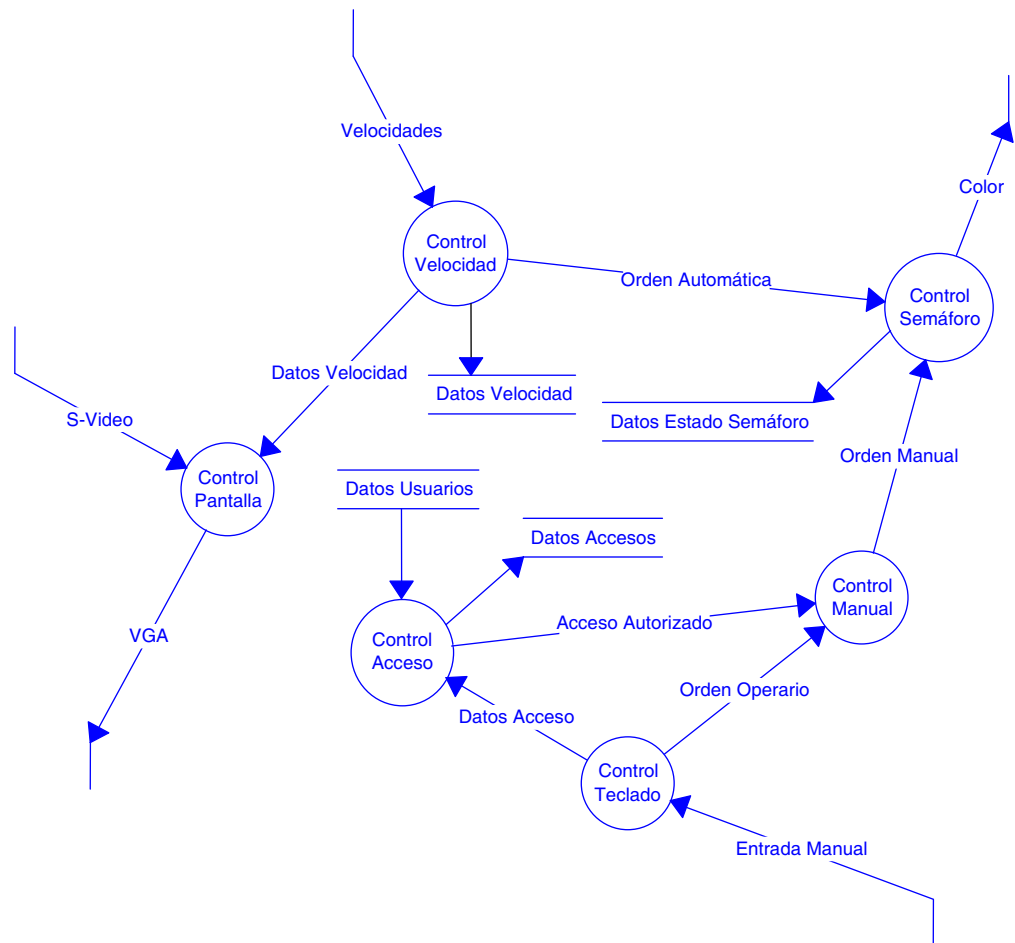
Entregue la hoja de lectura óptica con sus datos junto con su examen.

la entrada del túnel, de forma que cuando se detecte un coche circulando a una velocidad inferior a una estipulada, el semáforo se pondrá en rojo y cuando las velocidades detectadas por los sensores sean superiores a otro valor estipulado, se volverá a poner en verde.

Existirá un centro de control con una pantalla que muestre el estado del semáforo y las velocidades instantáneas captadas por el conjunto de sensores. El sistema podrá funcionar en modo manual, para lo cual el operario deberá introducir por teclado su identificador y contraseña cada vez que quiera pasar de automático a manual, o viceversa, y/o cambiar el estado del semáforo. Una o varias cámaras de video enviarán imágenes del interior y exterior del túnel, que se mostrarán en la pantalla y servirán para ayudar al operario a tomar decisiones cuando realice el control manual. El sistema almacenará todos los datos sobre velocidades proporcionadas por los sensores, el estado del semáforo, los cambios de estado que se produzcan en él y su origen (automático o manual) y los accesos de los usuarios, junto con la fecha y hora de cada evento.

DFD de nivel 1:

[Control Túnel]



Entregue la hoja de lectura óptica con sus datos junto con su examen.

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:

Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN

Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: NACIONAL 2ª SEMANA



Departamento de Ingeniería de
Software y Sistemas Informáticos

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En las preguntas teóricas, que se valoran con 2'5 puntos cada una, 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 notación de los DFD ¿a qué metodología de análisis está asociada? ¿Cómo se construye un modelo mediante esta notación? ¿Se le ocurre algún procedimiento para identificar los “procesos” y los “flujos de datos” del sistema?

SOLUCIÓN

La notación de los DFD está asociada al análisis estructurado (primer párrafo del epígrafe 4.2.3 “Diseño estructurado”, página 165).

Una descripción completa de cómo se construye el modelo, mediante esta notación, aparece en las páginas 52 a 57, epígrafe 2.3.2 “Diagramas de flujo de datos”.

Para identificar “procesos” y “flujos de datos” se podría utilizar Abbott (página 173), buscando identificar la información, de distinta naturaleza, que ‘fluye’ en distintas direcciones (flujos) y las descripciones de las acciones por las cuales se transforman esos flujos de información (los procesos).

2. Defina qué es un Tipo Abstracto de Datos (TAD) y deduzca las relaciones que se puedan establecer con los conceptos de “ocultación”, “genericidad”, “herencia” y “polimorfismo”.

SOLUCIÓN

Tradicionalmente, en la programación imperativa se optaba por separar los programas en dos partes: la de proceso y la de datos (ejemplos de lenguajes imperativos son FORTRAN, COBOL, PASCAL, BASIC...). La parte de proceso accedía y operaba directamente sobre los datos.

Esta separación produce una independencia funcional muy baja. Un ejemplo que ilustra esto es el “efecto 2000”, donde la simple adición de dígitos al formato de las fechas supuso realizar muchas modificaciones en la parte de proceso.

Para solventar estos problemas surgió el concepto de Tipo Abstracto de Datos (TAD), que agrupa en una sola entidad la representación de los datos y la parte de proceso que los manipula. Los

Entregue la hoja de lectura óptica con sus datos junto con su examen.

TADs ocultan la representación de los datos, que sólo es accesible desde las operaciones. De esta forma, un cambio en la representación de los datos de un TAD no se propaga a todo el programa, sino solamente a las operaciones del TAD.

Por tanto, la “ocultación” está garantizada en el TAD, gracias al propio concepto de esta abstracción que oculta la representación de los datos y sólo deja visible las operaciones que manipulan a dichos datos.

Por la propia naturaleza del TAD, se pueden definir tipos de datos que recogen una solución “genérica” o aplicable a varios casos particulares. Un tipo ‘genérico’ es un tipo parametrizable, es decir, que cubre un abanico de soluciones particulares y se adecua a distintos usos.

En cuanto a la “herencia”, un TAD es una abstracción y no está dotada del mecanismo de la herencia.

Por último, la capacidad representar algo o de actuar de distinta forma, sin que se pierda la propia naturaleza (polimorfismo), se puede alcanzar mediante la genericidad. Por tanto, un TAD puede ser polimorfo si es genérico; no es así con respecto al polimorfismo de herencia, del que no dispone un TAD por ser una abstracción.

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

3. Como resultado del diseño arquitectónico de una aplicación hemos caracterizado un módulo llamado ‘Gestión de fechas’, que realiza las siguientes operaciones sobre una lista de fechas:
- Buscar la más reciente.
 - Buscar la más antigua.
 - Imprimir todas las fechas que pertenezcan a cierto mes.

Para este módulo, proponga un diseño mediante refinamiento progresivo utilizando:

A. Funciones.

B. Abstracciones.

SOLUCIÓN

A. Tradicionalmente, la atención en el desarrollo del software se centraba en los aspectos funcionales, es decir, se trataba de responder a la cuestión ¿Cómo debe funcionar el programa para comportarse según lo acordado en el análisis? Entre las técnicas o metodologías de diseño en esta época se encuentra la descomposición funcional por refinamientos progresivos.

La descomposición funcional por refinamientos progresivos es una técnica de diseño que proviene de la metodología de programación, desarrollada en los años 60 y 70 principalmente por Dijkstra, y se conoce como programación estructurada. Su principal característica consiste en ver la aplicación compuesta de funciones o procedimientos donde entran unos datos y salen otros transformados. Para lograr esto de forma eficiente y estructurada se diseña atendiendo a las siguientes directrices:

- La especificación de las funciones se expresa mediante un conjunto de construcciones lógicas, las cuales deben garantizar la realización de cualquier tarea y deben ser suficientemente formales para evitar ambigüedades. Para ello se propone como constructores la *secuencia*, la *condición* y la *repetición* empleando para ello un lenguaje en pseudocódigo.
- Se aplica el refinamiento progresivo para el desarrollo de la aplicación, en el siguiente sentido: se plantea inicialmente el programa como una única función global y se va descomponiendo (refinando) poco a poco en funciones más sencillas.
- Las estructuras de datos, importantes en cualquier diseño, deben estar supeditadas a la funcionalidad de la aplicación. Estas funciones, utilizan los datos como elementos de

Entregue la hoja de lectura óptica con sus datos junto con su examen.

entrada para su utilización o transformación; esto se puede conseguir haciendo que los datos sean variables globales (son vistas desde cualquier función, aunque se pierda cohesión) o se pasan como parámetros de entrada. Estas estructuras de datos se pueden diseñar en cualquier momento y con independencia de la parte operacional.

Primero empezamos planteando el módulo Gestión de Fechas como una función global e iremos descomponiéndola en otras funciones. Para especificar el comportamiento de estas funciones utilizamos pseudocódigo:

Gestión de fechas →

```
CASO opcion de menu
  SI_ES fecha reciente ENTONCES BuscarFechaReciente
  SI_ES fecha antigua ENTONCES BuscarFechaAntigua
  SI_ES listar fecha del mes ENTONCES ImprimirFechaMes
```

BuscarFechaReciente →

```
FechaInicial = Primer elemento de la Lista de Fechas
PARA_CADA Elemento EN Lista de Fechas HACER
  SI FechaInicial es MENOR que Elemento ENTONCES
    FechaInicial = Elemento
FIN-SI
FIN-PARA
```

...

B. Posteriormente, se comenzó a otorgar mayor relevancia a los datos. Es decir, el desarrollo de una aplicación pasó a enfocarse como: ¿qué conceptos o datos intervienen en el programa?, ¿cuáles son sus operaciones o responsabilidades? y ¿cómo se relacionan entre sí?

En un diseño basado en abstracciones, antes de realizar la descomposición funcional u operativa de la aplicación deben quedar identificadas las posibles abstracciones (tipos abstractos de datos, funciones o bien datos encapsulados) que aparecerán en nuestro diseño. Para ello se siguen los siguientes pasos:

- i. Identificar claramente las abstracciones que aparezcan en la aplicación. Esto se puede hacer, a partir de las especificaciones, utilizando, por ejemplo, el método de búsqueda de Abbott.
- ii. Definir las operaciones asociadas a cada abstracción.
- iii. Describir las operaciones mediante lenguaje natural.
- iv. A partir de las especificaciones del programa llevar a cabo una descomposición funcional como la descrita en el apartado anterior. Así mismo reemplazar las descripciones de las operaciones de cada abstracción mediante construcciones estructuradas.

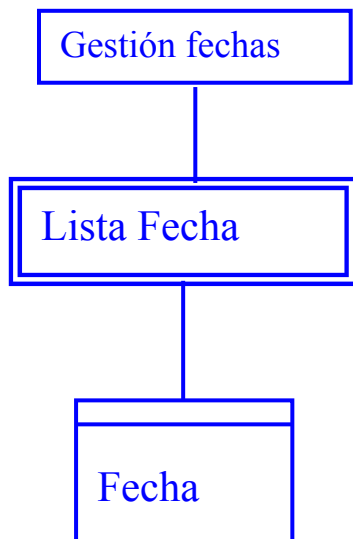
Podemos ver el módulo Gestión de fechas como una abstracción funcional global que gestiona a todas las demás abstracciones.

Podemos identificar la lista de fechas como un tipo abstracto de datos, ahora bien, como sólo vamos a tener una lista, podríamos considerarlo un tipo encapsulado. A esta lista estarían asociadas las operaciones de buscar fechas. Otro tipo de datos abstractos sería el que encapsularía tipo fecha y podríamos asociarle, entre otras operaciones, la de comparar con otra fecha.

Dato: Lista fecha (Dato encapsulado)
Atributos
 lista de elementos del tipo fecha
Operaciones
 Buscar fecha reciente
 Buscar fecha antigua
 Listar fecha mes

Dato: Fecha (Tipo Abstracto de datos)
Atributos
 Año
 Mes
 Día
Operaciones
 Comparar con otra fecha

DIAGRAMA DE ABSTRACCIONES



ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:

Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN

Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: EXTRANJERO ORIGINAL
(Europa Continental y Guinea)



Departamento de Ingeniería de
Software y Sistemas Informáticos

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En las preguntas teóricas, que se valoran con 2'5 puntos cada una, 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)

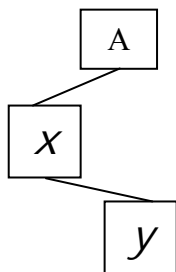
PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. ¿Qué punto de vista ofrecen los diagramas de transición de estados en la comprensión del sistema?

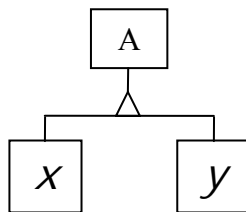
SOLUCIÓN

La evolución temporal del comportamiento deseado para el sistema. Epígrafe 2.3.3, primer y segundo párrafo del epígrafe, en las páginas 57 y 58.

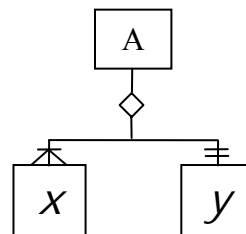
2. Enumere y explique las relaciones entre bloques que se representan en estos diagramas de diseño. ¿Se puede producir 'sobrecarga' en el módulo 'x' del diagrama 'C')?



A)



B)



C)

SOLUCIÓN

La notación corresponde a 'diagramas de estructura'; utilizados como notación de diseño, principalmente, en el diseño arquitectónico.

Entregue la hoja de lectura óptica con sus datos junto con su examen.

En el caso A), se representan relaciones de uso: ‘A’ llama o utiliza el módulo ‘x’ y, éste, invoca o utiliza el módulo ‘y’.

En el caso B), se representan relaciones de herencia (que sólo se dan en los objetos): ‘A’ es el antecedente o ‘padre’ de los módulos ‘x’ e ‘y’, que son sus hijos. Por tanto, heredarán las propiedades estáticas y dinámicas del ‘padre’. Es en los hijos donde se puede producir *especialización*, adecuando las propiedades heredadas al uso especializado del ‘hijo’ o incorporando nuevas.

En el caso C), se representan relaciones de composición o agregación: ‘A’ está compuesto por una cantidad (se representa la cardinalidad de la relación) de instancias del módulo ‘x’ y por un número de instancias del módulo ‘y’. La *sobrecarga* es una manera de expresar el polimorfismo que está vinculado, estrechamente, con el mecanismo de la herencia y sólo se produce con la parte dinámica de los objetos (‘metodos’). Como lo que se representa en el diagrama es la composición y no la herencia, no se puede decir si existe la posibilidad de sobrecarga en este módulo concreto. Podría producirse sobrecarga, si el módulo ‘x’ fuera una clase u objeto que heredara de otro; en cuyo caso, aparecería en otro diagrama distinto en el que se representaría la herencia.

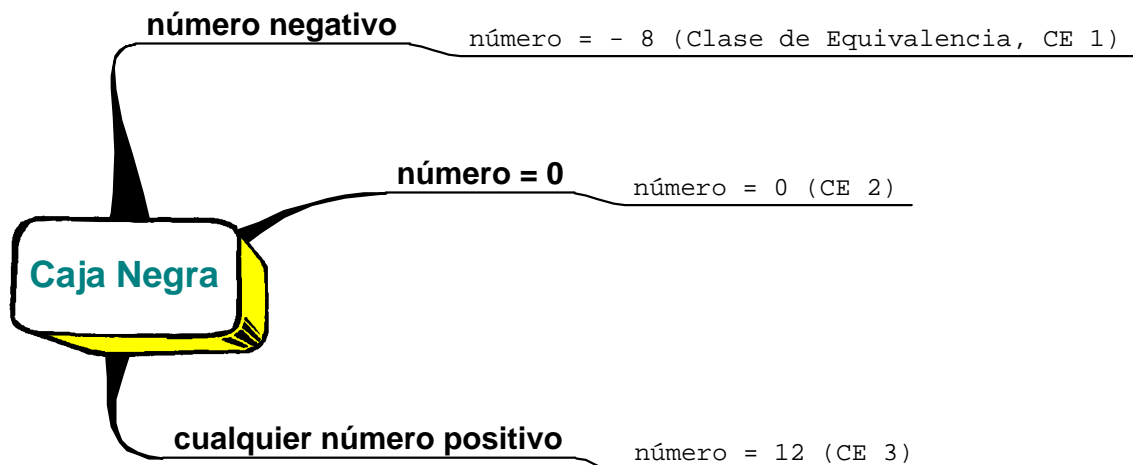
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Dado el subprograma `Factorial` escrito en Modula 2, que calcula el factorial de un número entero positivo y devuelve el valor **-1** en caso de ser negativo, verifique su funcionamiento correcto con pruebas:
- De caja negra: determine las clases de equivalencia necesarias e implemente un subprograma llamado `TestsCajaNegra` que ejecute las pruebas correspondientes.
 - De caja transparente: implemente un subprograma llamado `TestsCajaTransparente` que pruebe el cubrimiento lógico y el bucle (cuyo número de repeticiones no está acotado).

Nota: la codificación de los subprogramas pedidos puede realizarse en pseudocódigo o en Modula-2.

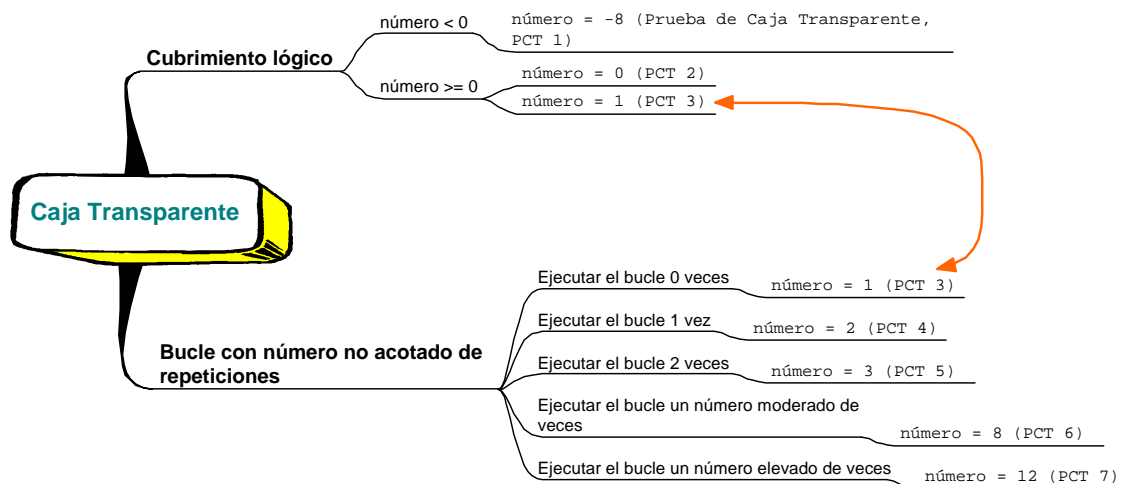
```
PROCEDURE Factorial(numero: INTEGER): INTEGER;
VAR
    resultado: INTEGER;
BEGIN
    IF numero < 0 THEN
        RETURN -1;
    END;
    resultado := 1;
    WHILE numero > 1 DO
        resultado := resultado * numero;
        DEC(numero);
    END;
    RETURN resultado;
END Factorial;
```

Pruebas de caja negra:



```
PROCEDURE TestsCajaNegra;
BEGIN
    IF Factorial(-8) <> -1 THEN
        WriteString("Error en la CE 1");
        WriteLn;
    ELSIF Factorial(0) <> 1 THEN
        WriteString("Error en la CE 2");
        WriteLn;
    ELSIF Factorial(12) <> 479001600 THEN
        WriteString("Error en la CE 3");
        WriteLn;
    END;
END TestsCajaNegra;
```

Pruebas de caja transparente:



```
PROCEDURE TestsCajaTransparente;
BEGIN
    IF Factorial(-8) <> -1 THEN
        WriteString("Error en la PCT 1");
        WriteLn;
    ELSIF Factorial(0) <> 1 THEN
        WriteString("Error en la PCT 2");
        WriteLn;
    ELSIF Factorial(1) <> 1 THEN
        WriteString("Error en la PCT 3");
        WriteLn;
    ELSIF Factorial(2) <> 2 THEN
        WriteString("Error en la PCT 4");
        WriteLn;
    ELSIF Factorial(3) <> 6 THEN
        WriteString("Error en la PCT 5");
        WriteLn;
    ELSIF Factorial(8) <> 40320 THEN
        WriteString("Error en la PCT 6");
        WriteLn;
    ELSIF Factorial(12) <> 479001600 THEN
        WriteString("Error en la PCT 7");
        WriteLn;
    END;
END TestsCajaTransparente;
```

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:

Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN

Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: EXTRANJERO ORIGINAL
(Londres, Lisboa y Tánger)



Departamento de Ingeniería de
Software y Sistemas Informáticos

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En las preguntas teóricas, que se valoran con 2'5 puntos cada una, 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 notación de los DFD ¿a qué metodología de análisis está asociada? ¿Cómo se construye un modelo mediante esta notación? ¿Se le ocurre algún procedimiento para identificar los “procesos” y los “flujos de datos” del sistema?

SOLUCIÓN

La notación de los DFD está asociada al análisis estructurado (primer párrafo del epígrafe 4.2.3 “Diseño estructurado”, página 165).

Una descripción completa de cómo se construye el modelo, mediante esta notación, aparece en las páginas 52 a 57, epígrafe 2.3.2 “Diagramas de flujo de datos”.

Para identificar “procesos” y “flujos de datos” se podría utilizar Abbott (página 173), buscando identificar la información, de distinta naturaleza, que ‘fluye’ en distintas direcciones (flujos) y las descripciones de las acciones por las cuales se transforman esos flujos de información (los procesos).

2. Defina qué es un Tipo Abstracto de Datos (TAD) y deduzca las relaciones que se puedan establecer con los conceptos de “ocultación”, “genericidad”, “herencia” y “polimorfismo”.

SOLUCIÓN

Tradicionalmente, en la programación imperativa se optaba por separar los programas en dos partes: la de proceso y la de datos (ejemplos de lenguajes imperativos son FORTRAN, COBOL, PASCAL, BASIC...). La parte de proceso accedía y operaba directamente sobre los datos.

Esta separación produce una independencia funcional muy baja. Un ejemplo que ilustra esto es el “efecto 2000”, donde la simple adición de dígitos al formato de las fechas supuso realizar muchas modificaciones en la parte de proceso.

Entregue la hoja de lectura óptica con sus datos junto con su examen.

Para solventar estos problemas surgió el concepto de Tipo Abstracto de Datos (TAD), que agrupa en una sola entidad la representación de los datos y la parte de proceso que los manipula. Los TADs ocultan la representación de los datos, que sólo es accesible desde las operaciones. De esta forma, un cambio en la representación de los datos de un TAD no se propaga a todo el programa, sino solamente a las operaciones del TAD.

Por tanto, la “ocultación” está garantizada en el TAD, gracias al propio concepto de esta abstracción que oculta la representación de los datos y sólo deja visible las operaciones que manipulan a dichos datos.

Por la propia naturaleza del TAD, se pueden definir tipos de datos que recogen una solución “genérica” o aplicable a varios casos particulares. Un tipo ‘genérico’ es un tipo parametrizable, es decir, que cubre un abanico de soluciones particulares y se adecua a distintos usos.

En cuanto a la “herencia”, un TAD es una abstracción y no está dotada del mecanismo de la herencia.

Por último, la capacidad representar algo o de actuar de distinta forma, sin que se pierda la propia naturaleza (polimorfismo), se puede alcanzar mediante la genericidad. Por tanto, un TAD puede ser polimorfo si es genérico; no es así con respecto al polimorfismo de herencia, del que no dispone un TAD por ser una abstracción.

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

3. Como resultado del diseño arquitectónico de una aplicación hemos caracterizado un módulo llamado ‘Gestión de fechas’, que realiza las siguientes operaciones sobre una lista de fechas:
- Buscar la más reciente.
 - Buscar la más antigua.
 - Imprimir todas las fechas que pertenezcan a cierto mes.

Para este módulo, proponga un diseño mediante refinamiento progresivo utilizando:

A. Funciones.

B. Abstracciones.

SOLUCIÓN

A. Tradicionalmente, la atención en el desarrollo del software se centraba en los aspectos funcionales, es decir, se trataba de responder a la cuestión ¿Cómo debe funcionar el programa para comportarse según lo acordado en el análisis? Entre las técnicas o metodologías de diseño en esta época se encuentra la descomposición funcional por refinamientos progresivos.

La descomposición funcional por refinamientos progresivos es una técnica de diseño que proviene de la metodología de programación, desarrollada en los años 60 y 70 principalmente por Dijkstra, y se conoce como programación estructurada. Su principal característica consiste en ver la aplicación compuesta de funciones o procedimientos donde entran unos datos y salen otros transformados. Para lograr esto de forma eficiente y estructurada se diseña atendiendo a las siguientes directrices:

- La especificación de las funciones se expresa mediante un conjunto de construcciones lógicas, las cuales deben garantizar la realización de cualquier tarea y deben ser suficientemente formales para evitar ambigüedades. Para ello se propone como constructores la *secuencia*, la *condición* y la *repetición* empleando para ello un lenguaje en pseudocódigo.
- Se aplica el refinamiento progresivo para el desarrollo de la aplicación, en el siguiente sentido: se plantea inicialmente el programa como una única función global y se va descomponiendo (refinando) poco a poco en funciones más sencillas.

Entregue la hoja de lectura óptica con sus datos junto con su examen.

- Las estructuras de datos, importantes en cualquier diseño, deben estar supeditadas a la funcionalidad de la aplicación. Estas funciones, utilizan los datos como elementos de entrada para su utilización o transformación; esto se puede conseguir haciendo que los datos sean variables globales (son vistas desde cualquier función, aunque se pierda cohesión) o se pasan como parámetros de entrada. Estas estructuras de datos se pueden diseñar en cualquier momento y con independencia de la parte operacional.

Primero empezamos planteando el módulo Gestión de Fechas como una función global e iremos descomponiéndola en otras funciones. Para especificar el comportamiento de estas funciones utilizamos pseudocódigo:

Gestión de fechas →

```
CASO opcion de menu
  SI_ES fecha reciente ENTONCES BuscarFechaReciente
  SI_ES fecha antigua ENTONCES BuscarFechaAntigua
  SI_ES listar fecha del mes ENTONCES ImprimirFechaMes
```

BuscarFechaReciente →

```
FechaInicial = Primer elemento de la Lista de Fechas
PARA_CADA Elemento EN Lista de Fechas HACER
  SI FechaInicial es MENOR que Elemento ENTONCES
    FechaInicial = Elemento
  FIN-SI
FIN-PARA
```

...

B. Posteriormente, se comenzó a otorgar mayor relevancia a los datos. Es decir, el desarrollo de una aplicación pasó a enfocarse como: ¿qué conceptos o datos intervienen en el programa?, ¿cuáles son sus operaciones o responsabilidades? y ¿cómo se relacionan entre sí?

En un diseño basado en abstracciones, antes de realizar la descomposición funcional u operativa de la aplicación deben quedar identificadas las posibles abstracciones (tipos abstractos de datos, funciones o bien datos encapsulados) que aparecerán en nuestro diseño. Para ello se siguen los siguientes pasos:

- i. Identificar claramente las abstracciones que aparezcan en la aplicación. Esto se puede hacer, a partir de las especificaciones, utilizando, por ejemplo, el método de búsqueda de Abbott.
- ii. Definir las operaciones asociadas a cada abstracción.
- iii. Describir las operaciones mediante lenguaje natural.
- iv. A partir de las especificaciones del programa llevar a cabo una descomposición funcional como la descrita en el apartado anterior. Así mismo reemplazar las descripciones de las operaciones de cada abstracción mediante construcciones estructuradas.

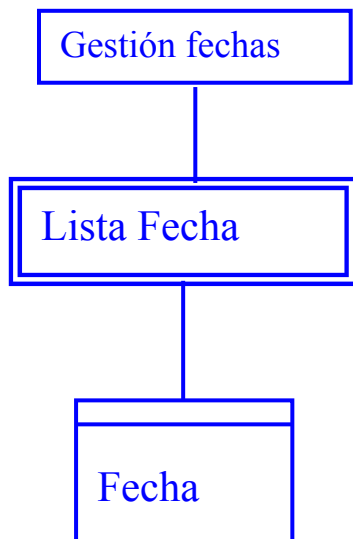
Podemos ver el módulo Gestión de fechas como una abstracción funcional global que gestiona a todas las demás abstracciones.

Podemos identificar la lista de fechas como un tipo abstracto de datos, ahora bien, como sólo vamos a tener una lista, podríamos considerarlo un tipo encapsulado. A esta lista estarían asociadas las operaciones de buscar fechas. Otro tipo de datos abstractos sería el que encapsularía tipo fecha y podríamos asociarle, entre otras operaciones, la de comparar con otra fecha.

Dato: Lista fecha (Dato encapsulado)
Atributos
 lista de elementos del tipo fecha
Operaciones
 Buscar fecha reciente
 Buscar fecha antigua
 Listar fecha mes

Dato: Fecha (Tipo Abstracto de datos)
Atributos
 Año
 Mes
 Día
Operaciones
 Comparar con otra fecha

DIAGRAMA DE ABSTRACCIONES



ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:

Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN

Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: AMÉRICA ORIGINAL



Departamento de Ingeniería de
Software y Sistemas Informáticos

Todas las preguntas de este ejercicio son eliminatorias en el sentido de que debe obtener una nota mínima en cada una de ellas. En las preguntas teóricas, que se valoran con 2'5 puntos cada una, 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. Enumere y comente los problemas e inconvenientes que encuentra en el 'ciclo de vida en espiral'.

SOLUCIÓN

La fase de 'Análisis de Riesgos' se ejecuta, normalmente, por una autoridad con experiencia. Esta fase consiste en identificar las fuentes y causas de riesgo, determinar las consecuencias de cada una sobre el desarrollo del producto, establecer un plan de actuación para el caso de que se produzca la situación de riesgo e incorporar la planificación al plan general del desarrollo. Este ciclo de vida incorpora tareas propias de la gestión del desarrollo (planificación y análisis de riesgo) y, por ser mucho más completo y sofisticado que otros modelos, requiere una elaboración más minuciosa. Por otro lado, el análisis de riesgo está estrechamente vinculado a la planificación de la parte que se va a afrontar. Por tanto, la forma de desarrollar el producto se basa, en gran parte, en la propia experiencia del analista de riesgos. Esto puede ser especialmente crítico en las frecuentes ocasiones en las que el analista se ve obligado a 'convencer' al cliente de porqué se planifica y se afronta una parte del desarrollo y no otra.

2. ¿Qué es la validación? ¿Por qué deben incluirse los requisitos de validación en el SRD?

SOLUCIÓN

La 'validación' consiste en la comprobación de que el producto o uno de sus elementos, satisface las especificaciones establecidas por el cliente en la fase de análisis (pág. 16). Normalmente, esas "especificaciones establecidas por el cliente" son las condiciones que él pone para aceptar el producto. Debido al carácter contractual de las mencionadas condiciones de validación, es por lo que resulta muy conveniente que se expresen en documento SRD.

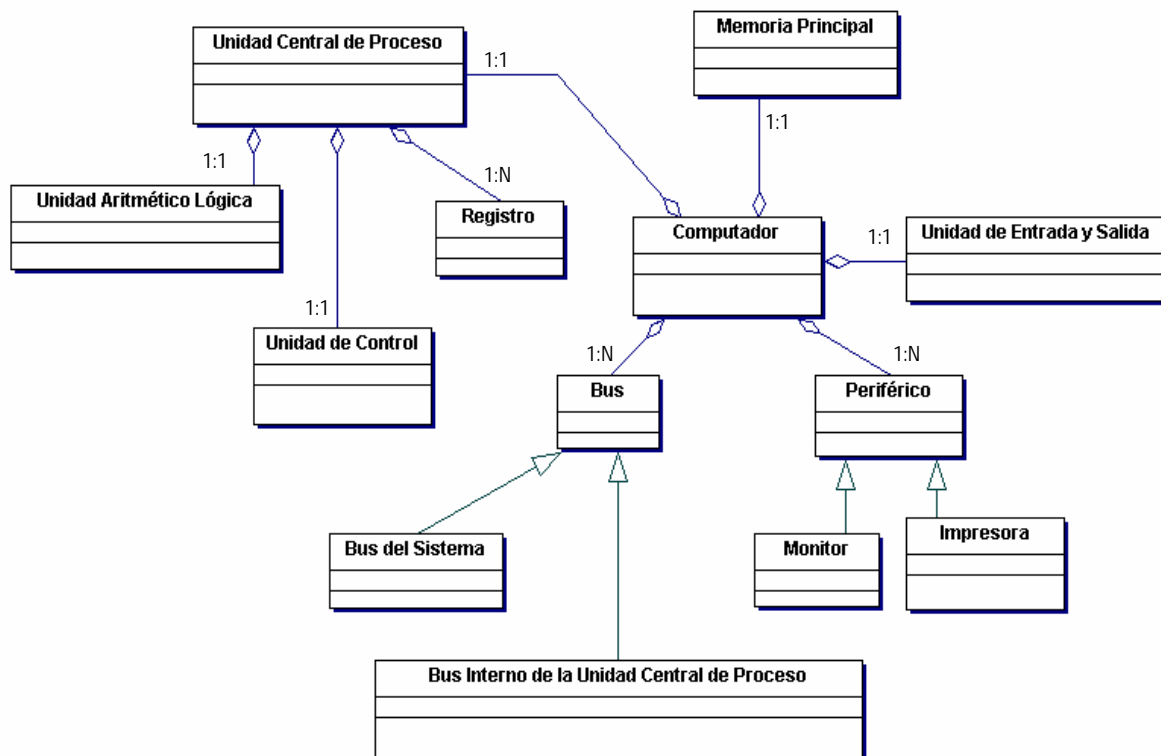
Entregue la hoja de lectura óptica con sus datos junto con su examen.

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

3. Según la arquitectura Von Neumann, un computador está compuesto por:
1. Unidad Central de Proceso: controla el funcionamiento del computador y lleva a cabo sus funciones de procesamiento de datos. A su vez, consta de:
 - a. Unidad Aritmético Lógica: se encarga de realizar operaciones elementales como la suma, resta, AND...
 - b. Unidad de Control: se encarga de leer y ejecutar las instrucciones máquina almacenadas en la memoria principal.
 - c. Registros: proporcionan almacenamiento interno a la Unidad Central de Proceso.
 2. Memoria Principal.
 3. Unidad de Entrada y Salida: transfiere datos entre el computador y los periféricos.
 4. Periféricos: los hay de distinto tipo, por ejemplo,
 - a. Monitor
 - b. Impresora
 5. Elementos de interconexión o buses: los hay de distinto tipo, por ejemplo,
 - a. Bus del Sistema: conecta Unidad Central de Proceso, Memoria Principal y Unidad de E/S.
 - b. Bus Interno de la Unidad Central de Proceso: conecta Unidad Aritmético Lógica, Unidad de Control y Registros.

Desarrolle un modelo de análisis para esta arquitectura Von Neumann. Para ello, utilice la notación de los ‘Diagramas de Objetos’ que se explica en el texto de la asignatura. No es necesario que incluya los atributos ni las operaciones de las clases. Límitese a representar las relaciones de herencia y composición entre los bloques funcionales descritos anteriormente.

SOLUCIÓN



Entregue la hoja de lectura óptica con sus datos junto con su examen.

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:

Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN

Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: NACIONAL ORIGINAL



Departamento de Ingeniería de
Software y Sistemas Informáticos

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.

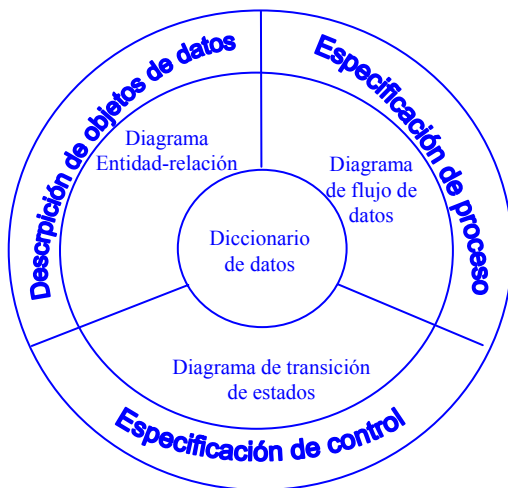
¡ATENCIÓN! PONGA SUS DATOS EN LA HOJA DE LECTURA ÓPTICA QUE DEBERÁ ENTREGAR JUNTO CON EL RESTO DE LAS RESPUESTAS.

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. Explique qué matices y puntos de vista proyectan sobre el modelo de un mismo sistema: los Diagramas de Flujos de Datos, los Diagramas Entidad - Relación, los Diagramas de Transición de Estados y el Diccionario de Datos.

SOLUCIÓN



Las tres primeras notaciones establecen tres puntos de vista, diferenciados y complementarios, sobre una misma cosa: el modelo del sistema, esto es, la descripción del producto que se va a construir. Como se ve en sinóptico de la izquierda, los diagramas de flujo de datos, aportan el punto de vista de la especificación de los procesos y transformaciones que se producen en la información que fluye. Los diagramas E-R, aportan la descripción de los elementos conceptuales y objetos (entidades) que se manejan en el sistema y las relaciones relevantes que existen entre ellos. Los diagramas de transición de estados, establecen la especificación del control que se establece el sistema; aportan la variable temporal, la evolución de él. Por último, el diccionario de datos

complementa y detalla las especificaciones realizadas con cualquiera de las notaciones anteriores, bien sea con lenguajes de descripción funcional o bien con descripciones de datos con cierto formalismo.

Entregue la hoja de lectura óptica con sus datos junto con su examen.

2. Dadas dos implementaciones alternativas de un algoritmo, ¿coincidirán las pruebas de **caja negra** que verifiquen el funcionamiento correcto de cada implementación? ¿Coincidirán las pruebas si son de **caja transparente**?

SOLUCIÓN

La estrategia de pruebas de caja negra, ignora por completo la estructura interna del programa y se basa, exclusivamente, en la comprobación de la especificación software. En este caso, el **juego de casos de prueba** al que se someten las dos versiones **será el mismo**. En esto consiste el método de ‘comparación de versiones’ para las pruebas de caja negra. Por el contrario, en la estrategia de caja transparente, los casos de prueba tratan de conseguir que el programa transite por todos los posibles caminos de ejecución y que se pongan en juego todos los elementos del código (que es conocido). Por ello, al depender (los casos de prueba) del código, dichos casos de prueba pueden ser distintos en cada implementación.

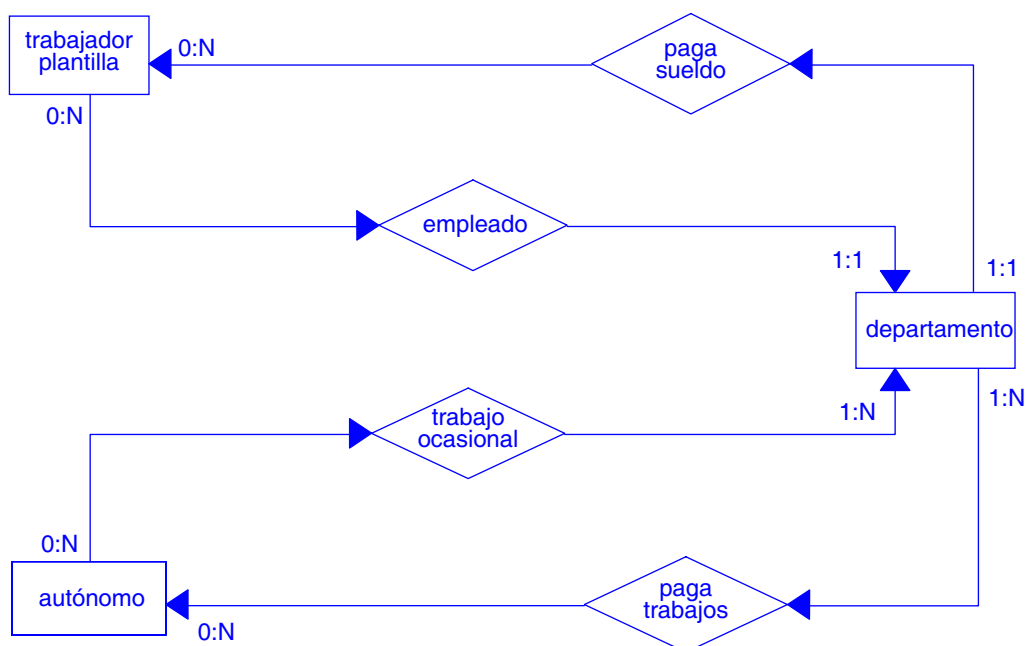
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Se desea modelar la aplicación de cálculo de nóminas de una empresa según la siguiente descripción:

En la empresa existen trabajadores de plantilla y trabajadores autónomos. Un trabajador de plantilla no puede pertenecer a más de un departamento de la empresa y tiene asignado un sueldo bruto anual. Los autónomos, que realizan trabajos ocasionales para la empresa, pueden hacerlo en distintos departamentos y cobrarán por cada trabajo realizado. Cada departamento dispone de su propio presupuesto para pagar a los trabajadores.

Realice el análisis, mediante diagrama entidad/relación, de la aplicación descrita.

SOLUCIÓN



Entregue la hoja de lectura óptica con sus datos junto con su examen.

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:

Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN

Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: NACIONAL RESERVA



Departamento de Ingeniería de
Software y Sistemas Informáticos

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. Dé una breve definición de la fase de diseño. Describa cuáles son las cualidades mínimas que se pretende alcanzar con la descomposición modular del diseño.

SOLUCIÓN

El diseño trata de definir y formalizar la estructura de un sistema informático con el suficiente detalle como para permitir su realización física (último párrafo de la página 103, epígrafe 3.1 “Introducción”). Las cualidades mínimas de la descomposición modular son la **independencia funcional**, la **comprensibilidad** y la **adaptabilidad** (epígrafes 4.1.1, 4.1.2 y 4.1.3 en página 150 y siguientes).

2. Enumere las diferencias y coincidencias entre las pruebas α y β

SOLUCIÓN

Coincidencias: son pruebas “de sistema” y se suele aplicar una estrategia de “caja negra”. Es el usuario el que realiza las pruebas.

Diferencias: en las pruebas alfa, el entorno de prueba está controlado (se registran las incidencias y se conocen las circunstancias en que se producen) y el usuario recibe el apoyo de alguna persona del equipo de desarrollo, el cual, puede seguir muy de cerca la evolución de las pruebas. En las pruebas beta, el sistema se prueba en el entorno normal de trabajo y sin apoyo de nadie involucrado en el desarrollo. El usuario es el encargado de transmitir, al equipo de desarrollo, las circunstancias en las que se han producido las incidencias (epígrafe 5.9.2 en páginas 290 y 291).

Entregue la hoja de lectura óptica con sus datos junto con su examen.

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

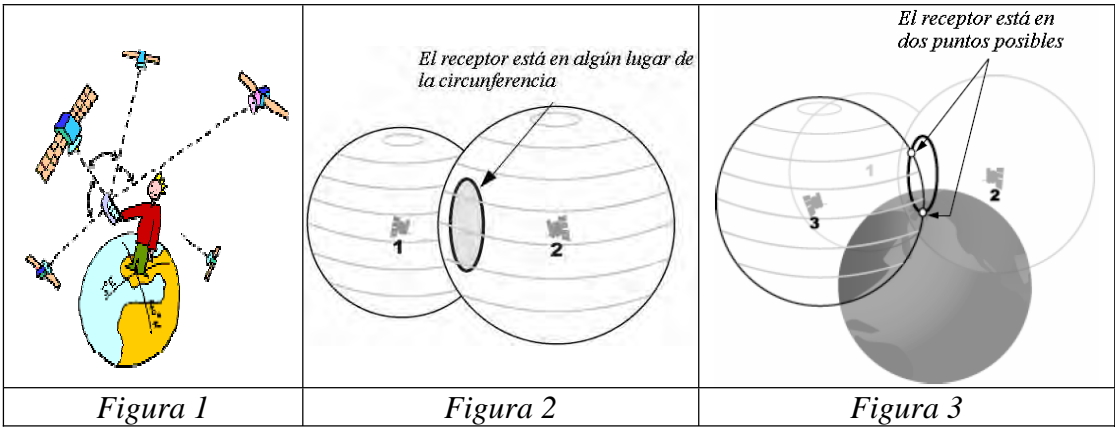
3. El Sistema de Posicionamiento Global (Global Positioning System, GPS) permite conocer a sus usuarios la posición que ocupan en el globo terráqueo (Figura 1). El sistema consta de, al menos, 24 satélites, 5 estaciones de control terrestres y aparatos receptores. Cada satélite emite periódicamente una señal con información sobre su localización en órbita y el instante en que se ha producido la emisión. Desde la Tierra, un receptor puede conocer su localización si recibe señal desde, al menos, 4 satélites. Conocido el instante en que se emitió una señal y la velocidad a la que viajan las ondas emitidas (la de la luz), el receptor puede calcular su distancia al satélite como:

$$\text{distancia} = (\text{tiempo de la recepción} - \text{tiempo de la emisión}) \times \text{velocidad de la luz}$$

Si un receptor recibe señal de un satélite, sabe que se encuentra en la superficie de una esfera de radio la distancia al satélite y centro la localización en órbita del satélite; si recibe señal de 2 satélites, se encuentra en la intersección de dos superficies esféricas, es decir, una circunferencia (Figura 2); si recibe de 3 satélites, las posibilidades se reducen a dos puntos (Figura 3); y, si la recibe de 4, el receptor conoce su posición exacta en el globo terráqueo.

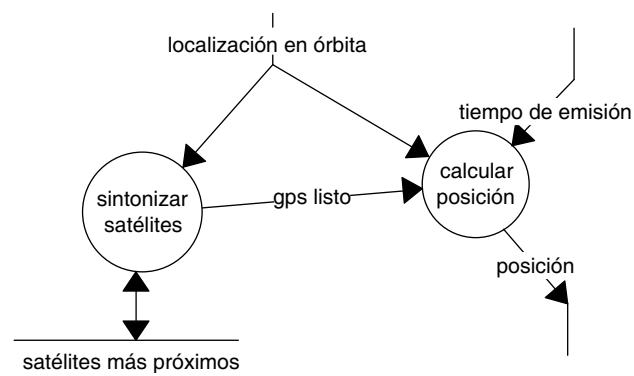
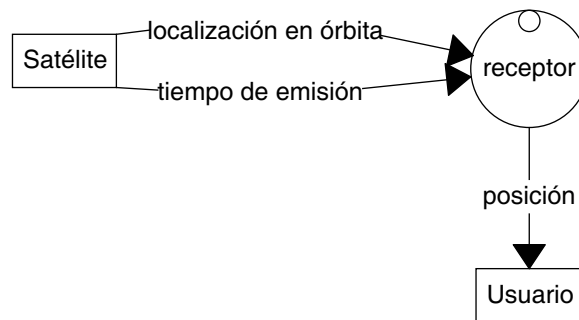
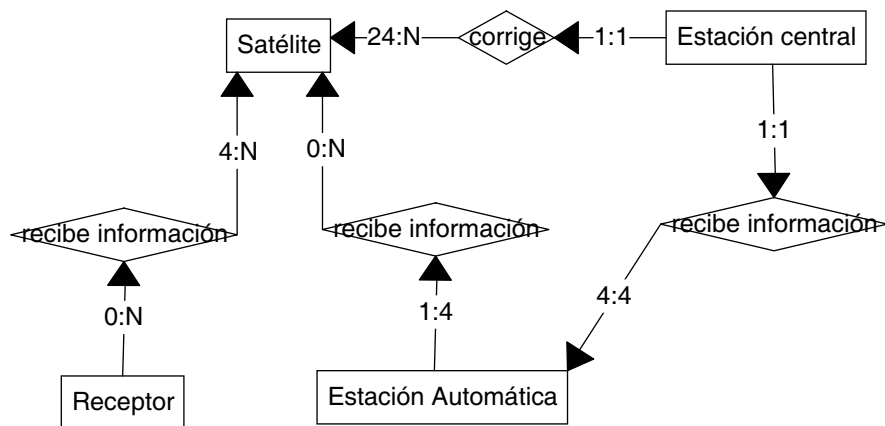
Normalmente, los receptores guardan un histórico con las localizaciones en órbita de los últimos satélites desde los que recibieron señal. De este modo, cuando se enciende un receptor, éste no inicia una búsqueda desde cero, sino que prueba con los satélites previamente almacenados (sintonizar).

La órbita que describe un satélite puede sufrir ligeras variaciones. Para mantener la precisión en la localización de los satélites, existen cinco estaciones de control terrestres que ajustan esta información. Concretamente, hay cuatro estaciones automáticas que reciben señales de los satélites y las envían a una estación central que procesa toda la información y transmite las correcciones pertinentes a los satélites.



Se pide:

- **Modelar la descripción anterior mediante un Diagrama Entidad-Relación**
- **Modelar con un Diagrama de Flujo de Datos el software de un receptor**



ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:

Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN

Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: NACIONAL 1ª SEMANA



Departamento de Ingeniería de
Software y Sistemas Informáticos

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. ¿Qué dimensión del proceso de desarrollo de software añade el modelo en V al modelo en cascada? ¿Qué implica respecto a la comprobación de la corrección en las distintas fases del ciclo de vida?

Solución

En el modelo en V se contempla **el nivel de detalle** sobre el que se trabaja en cada una de las fases. El SRD, documento producido en la fase de análisis, contempla el **sistema** en su totalidad. Tras la fase de diseño, en la que se lleva a cabo una descomposición del sistema para abordar su codificación, se desciende al nivel de **módulo**. Tras la codificación de todos los módulos se realiza su integración, ascendiéndose de nuevo al nivel de sistema completo.

El modelo en V pone de manifiesto que la salida de una fase del ciclo de vida no sólo afecta a la siguiente. Existe una relación entre salidas y entradas de fases no consecutivas en el tiempo pero situadas en el mismo nivel de detalle. Este hecho se observa a la hora de realizar comprobaciones de la corrección. La **validación** se hace tras la fase de integración, donde nos encontramos en el nivel del sistema completo, comprobándose que se cumple lo estipulado en el documento producido tras la fase de análisis (SRD). La **verificación** se lleva a cabo tras la fase de codificación, realizándose una comprobación de la corrección del sistema a nivel de módulo.

2. Defina pruebas alfa, beta, de caja negra y de caja transparente. Razone si las pruebas alfa y beta se pueden considerar como de caja negra o de caja transparente.

Solución

Las pruebas alfa y beta sirven para comprobar si un sistema completo satisface las especificaciones en un entorno real de trabajo. Mientras que las pruebas alfa se realizan en un entorno controlado donde el usuario tiene el apoyo de algún desarrollador, durante las pruebas beta no existe dicho apoyo.

Entregue la hoja de lectura óptica con sus datos junto con su examen.

Previas a las “pruebas de sistema”, se realizan las “pruebas de unidades”, que comprueban por separado cada módulo constituyente del sistema. Dentro de las pruebas de unidades pueden distinguirse las de “caja negra”, donde se ignora por completo la estructura interna de un módulo, y las de “caja transparente”, donde se conoce y tiene en cuenta dicha estructura.

En las pruebas alfa y beta los usuarios desconocen la estructura interna del sistema, luego pueden considerarse como de caja negra.

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

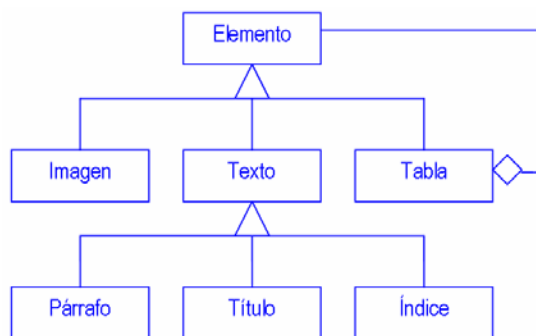
3. Se desea construir un procesador de textos capaz de manipular los siguientes elementos: imágenes, texto y tablas. En principio, sólo se contemplan tres tipos de texto: párrafos, títulos e índices. Por otro lado, una tabla puede albergar cualquier tipo de texto, imagen e incluso otras tablas.

Utilizando un diagrama de objetos (sin incluir las operaciones ni los atributos), elabore un diseño para los elementos que maneja el procesador de textos.

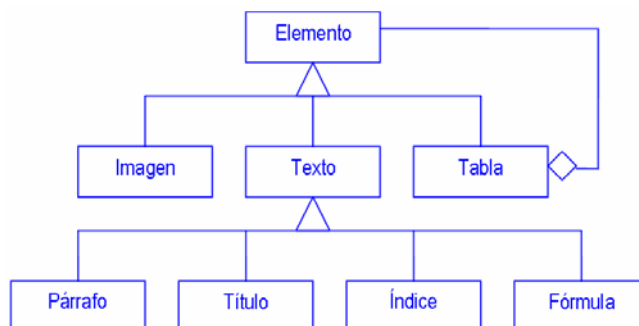
¿Cómo afectaría a su diseño la inclusión de fórmulas matemáticas como nuevo tipo de elemento de texto?

Solución

Diseño inicial:



Diseño contemplando “fórmulas matemáticas”:



Entregue la hoja de lectura óptica con sus datos junto con su examen.

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:

Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN

Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: NACIONAL 2ª SEMANA



Departamento de Ingeniería de
Software y Sistemas Informáticos

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. A. Describa la diferencia entre los factores de calidad de *corrección* y *fiabilidad*.

Solución

Corrección: Es el grado en que un producto software cumple con sus especificaciones. Podría estimarse como el porcentaje de requisitos que se cumplen adecuadamente.

Fiabilidad: Es el grado de ausencia de fallos durante la operación del producto software. Puede estimarse como el número de fallos producidos o el tiempo que permanece inutilizable durante un intervalo de operación dado.

Pág. 27 del libro

- B. Para evaluar la *corrección* de un sistema ¿qué tipo de prueba debería utilizarse, de caja negra o de caja transparente? Razone la respuesta.

Solución

Como se trata de evaluar si el sistema satisface sus especificaciones, lo correcto es hacer pruebas de caja negra, ya que se trataría de observar la respuesta del sistema (resultados) a determinadas entradas (casos de prueba) y comprobar que son los esperados.

Pág. 274 del libro

2. Explique brevemente los distintos niveles de comprobación de tipos que presentan los lenguajes de programación.

Solución

Véase pág. 255 y 256 del libro de texto.

Entregue la hoja de lectura óptica con sus datos junto con su examen.

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

3. Se ha recibido una petición, por parte de un usuario final, para desarrollar un sistema automático de acceso a un garaje. En ella dice:

“...Existe un único portón que sirve de entrada y salida, pero su ancho sólo permite el paso de un vehículo. Se instalarán sensores de peso que detecten la presencia de un vehículo y semáforos tanto en el interior como en el exterior del garaje. Para controlar el acceso se utilizarán mandos a distancia codificados.”

Analice el sistema y realice una descripción de su modelo utilizando lenguaje natural estructurado. Incluya los elementos que considere necesarios para controlar el acceso de vehículos autorizados así como para evitar daños físicos en los vehículos.

Realice el DFD de contexto del modelo descrito.

Solución

La “descripción del modelo” es un apartado fundamental del SRD. En él se define un modelo conceptual del sistema que se va a desarrollar. Este modelo establecerá las propiedades y restricciones del sistema, dando una visión de alto nivel sin descender a detalles concretos del mismo. Es decir, indicará QUÉ debe hacer, y no CÓMO lo debe hacer. Debe ser completo, conciso, sin ambigüedades, **sin detalles de diseño o implementación, fácilmente entendible por el cliente...** Para ello se puede emplear cualquiera de las notaciones para la especificación, siendo habitual el uso de varias de ellas, de forma que se facilite su entendimiento por parte de los que van a participar en el desarrollo del sistema.

En este caso, la descripción mediante lenguaje natural estructurado se complementará con el DFD de contexto, en el que se detallan los elementos (entidades externas) que facilitan datos o reciben órdenes del sistema software.

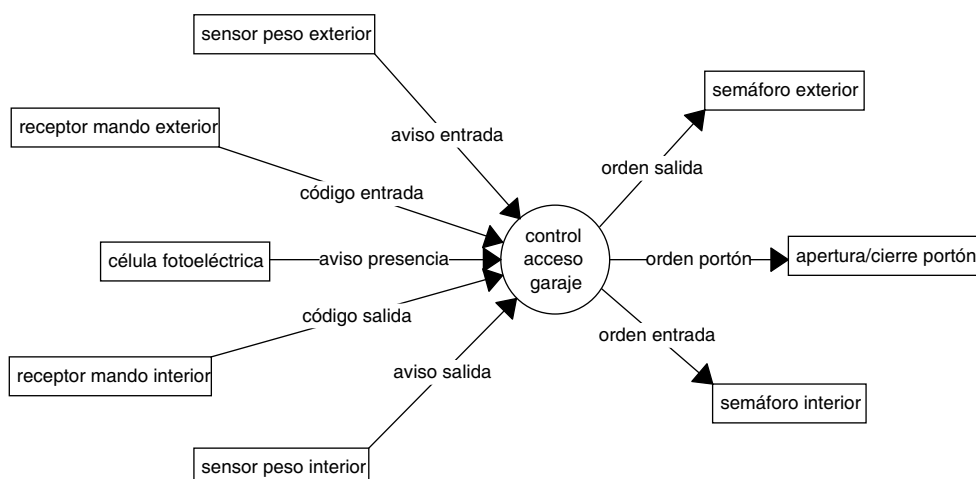
Sistema de control de acceso a garaje. Descripción del modelo

SI un coche autorizado quiere salir ENTONCES se pondrá en rojo el semáforo exterior, en verde el interior, y se abrirá el portón durante 30 segundos.

SI un coche autorizado quiere entrar ENTONCES se pondrá en rojo el semáforo interior, en verde el exterior, y se abrirá el portón durante 30 segundos.

SI la célula fotoeléctrica detecta un coche ENTONCES el portón permanecerá abierto durante 30 segundos.

SI un coche no autorizado quiere entrar o salir ENTONCES se pondrán en rojo los semáforos interior y exterior.



Entregue la hoja de lectura óptica con sus datos junto con su examen.

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:

Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN

Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: EUROPA RESERVA



Departamento de Ingeniería de
Software y Sistemas Informáticos

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. ¿Qué criterios se deben emplear para utilizar más de una notación en la construcción del modelo de análisis?

Solución

Sea cual sea la opción elegida, la notación o notaciones empleadas deberán ser fáciles de entender por el *cliente*, el usuario y, en general, por todos aquellos que puedan participar en el análisis o el desarrollo del sistema.

Si el objetivo 'global' del análisis es la comprensión del comportamiento del producto que se va a construir, la utilización de una o más notaciones para elaborar el modelo deberá decidirse de acuerdo a los criterios que debe cumplir el propio modelo:

- Completo y sin omisiones.
- Conciso y sin trivialidades.
- Sin ambigüedades.
- Sin detalles de diseño o implementación.
- Fácilmente entendible por el *cliente*.
- Separar los requisitos funcionales de los no funcionales.
- Dividido y jerarquizado.

2. Una aplicación de animación gráfica para la decoración y diseño de interiores utiliza un elemento denominado "Asiento". ¿Qué aspectos y mecanismos del diseño de software podría utilizar para la especialización de dicho elemento y su uso en un ambiente concreto (silla, butaca, banqueta, sofá, etc.)?

Solución

La especialización se puede conseguir mediante la abstracción y a través de la herencia. En este caso, lo más inmediato parece ser el uso del mecanismo de la herencia de manera que la especialización de cada instancia en su ambiente (silla, butaca, etc.) se consigue mediante el

Entregue la hoja de lectura óptica con sus datos junto con su examen.

polimorfismo hereditario, es decir, adecuando cada cuál sus características específicas que han heredado de la clase “Asiento”.

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

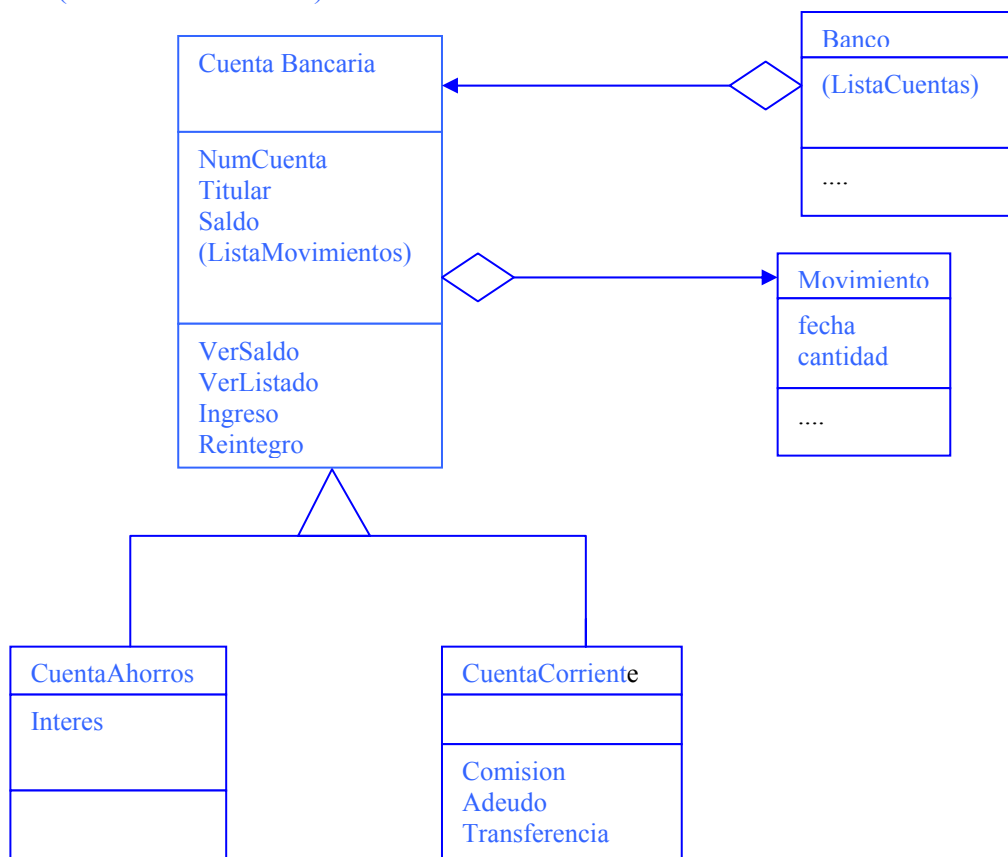
3. Un sistema informático de gestión bancaria opera con dos tipos de cuentas bancarias: la corriente y la de ahorros. Las cuentas disponen de un número que las identifica, tienen asociado un titular, una lista de los movimientos realizados hasta la fecha y almacenan el saldo disponible. Así mismo, las cuentas soportan operaciones como pedir el saldo y hacer ingresos o reintegros. Las cuentas de ahorros producen un interés que se calcula cada mes. Las cuentas corrientes no producen interés sino que reciben la carga de una comisión anual. Dichas cuentas corrientes facilitan diversas operaciones como adeudos domiciliarios y transferencias de dinero. Cada movimiento bancario consta de la fecha, la cantidad y el concepto de que se trata.

Modele la situación anterior utilizando un diseño orientado a objetos. Utilice herencia y composición cuando se pueda.

Solución

CLASES:

- Cuenta de ahorros
- Cuenta corriente
- Cuenta bancaria
- Movimientos
- Banco (Sistema de Gestión)



Entregue la hoja de lectura óptica con sus datos junto con su examen.

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:

Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN

Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: ORIGINAL AMÉRICA, GUINEA Y TÁNGER



Departamento de Ingeniería de
Software y Sistemas Informáticos

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. ¿Qué ventajas e inconvenientes tiene el uso de prototipos en el ciclo de vida y en qué situaciones se producen dichas ventajas e inconvenientes?

Solución

Véase apartado 1.5 del libro, en páginas 16 a 21.

2. ¿Se puede conseguir el polimorfismo sin utilizar la herencia? Si es así, ¿de qué manera?

Solución

“El concepto de genericidad (...) es una manera de lograr que un elemento genérico pueda adquirir distintas formas cuando se particulariza su utilización” (pág. 118 del libro).

Por otro lado, existe otro tipo de polimorfismo, que no está ligado a la herencia, en el que quienes adquieren múltiples formas son los operadores, funciones o procedimientos. Es el polimorfismo de **sobrecarga**.

Entregue la hoja de lectura óptica con sus datos junto con su examen.

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

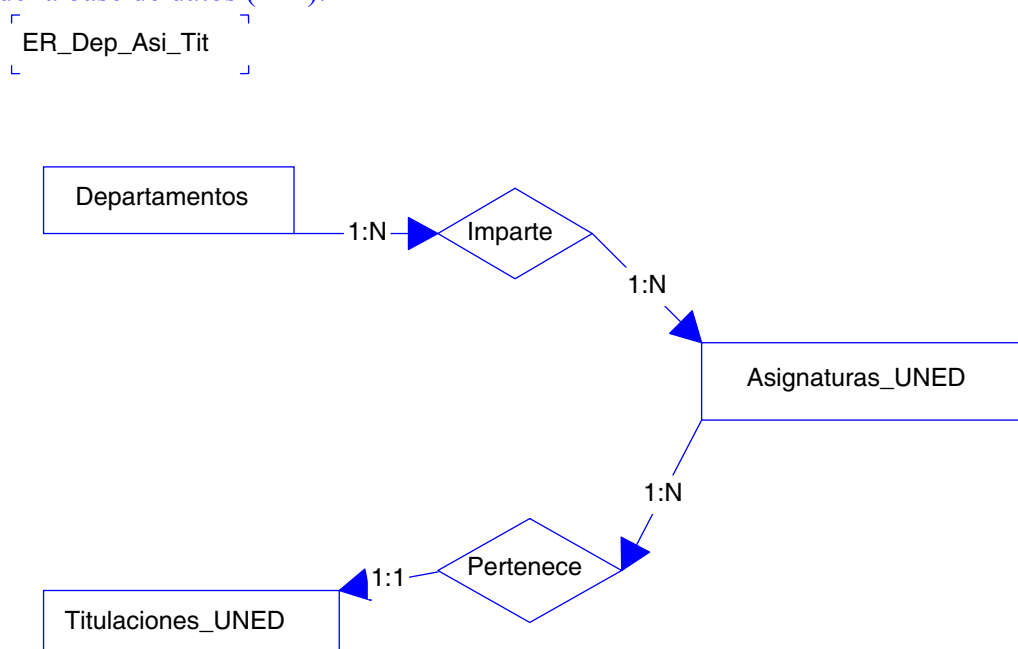
3. Se pretende hacer una aplicación para mejorar la gestión de convalidaciones. El procedimiento vigente es que el solicitante elabore un escrito con sus datos, la titulación de procedencia y la lista de asignaturas que quiere convalidar. Con esta información se construye un dossier en el que aparecen los datos mencionados. Según el tipo de convalidación solicitada (informativa, total, parcial, de titulación extranjera, etc.), cada dossier es revisado por el equipo docente correspondiente a cada asignatura cuya convalidación se solicita. Cada departamento imparte un conjunto de asignaturas de la titulación. Periódicamente, se reparten los dossieres entre los departamentos para la revisión de las asignaturas que les correspondan. El problema es que el listado de asignaturas no aparece ordenado ni agrupado por departamentos, lo cuál, dificulta el movimiento de las carpetas y su control.

La aplicación que se pretende desarrollar permitiría a un administrativo introducir los datos del solicitante, centro de procedencia, tipo de convalidación, titulación (Sistemas, Gestión ó 2º Ciclo) y seleccionar, en una lista, las asignaturas que se solicita convalidar. El resultado debe ser un documento en el que aparezcan impresos los datos anteriores y, lo más importante, las asignaturas agrupadas por departamentos, con el nombre del departamento que imparte cada grupo.

Realice el diseño de la base de datos con las titulaciones, departamentos y asignaturas. A continuación, construya un modelo de análisis, para la aplicación descrita, mediante DFD con dos o tres niveles.

Solución

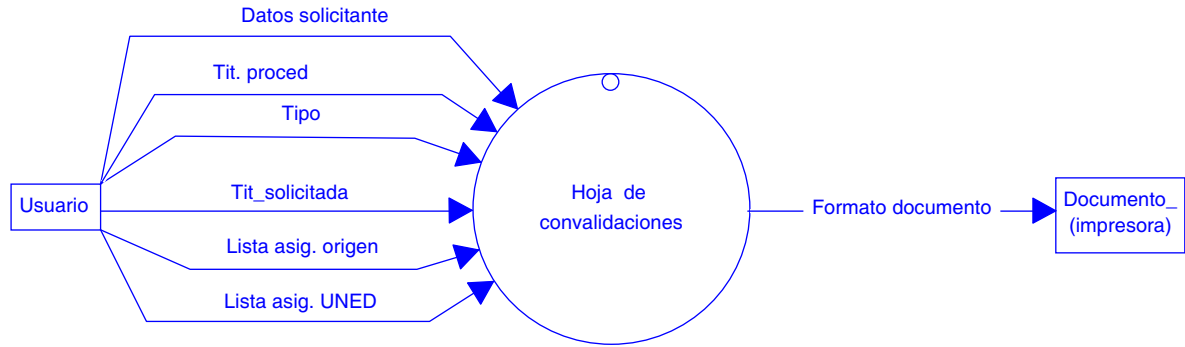
Diseño de la base de datos (E-R):



Entregue la hoja de lectura óptica con sus datos junto con su examen.

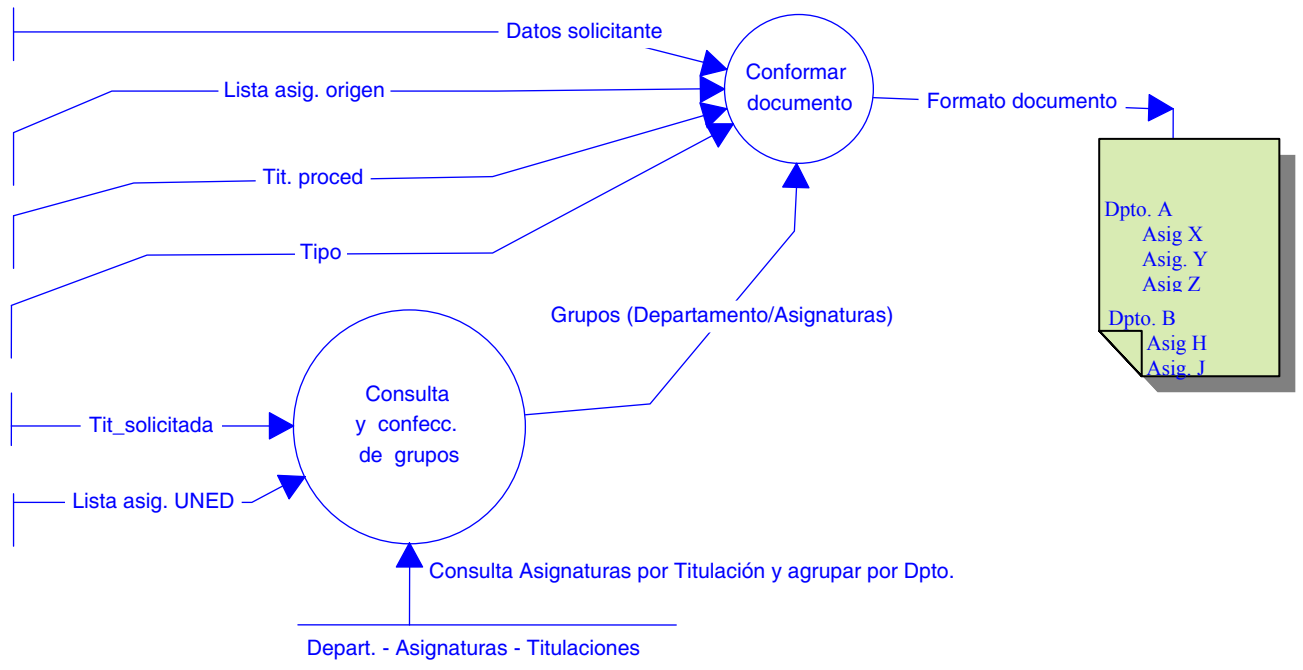
Modelado con DFD. Contexto (Nivel 0):

```
HdConvalidaciones
```



Nivel 1:

Hoja de convalidaciones



Entregue la hoja de lectura óptica con sus datos junto con su examen.

ASIGNATURA: INGENIERÍA DEL SOFTWARE (2º CURSO)

CÓDIGO DE ASIGNATURA: 210=SISTEMAS y 208=GESTIÓN

CÓDIGO CARRERA:

Plan de estudios en extinción: 40=SISTEMAS y 41=GESTIÓN

Plan de estudios NUEVO: 53=SISTEMAS y 54=GESTIÓN

MATERIAL PERMITIDO: NINGUNO

MODELO: RESERVA AMÉRICA, GUINEA Y TANGER



Departamento de Ingeniería de
Software y Sistemas Informáticos

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. Razone la causas por las que se descomponen los DFDs (Diagramas de Flujo de Datos). ¿Tendría sentido aplicar la descomposición en otras notaciones de análisis como DTEs (Diagramas de Transición de Estados) o DERs (Diagramas Entidad-Relación)?

Solución

Para mejorar la legibilidad del DFD de un sistema complejo, facilitar la construcción del DFD, permitir el desarrollo en paralelo del DFD e incluso la reutilización parcial del DFD, éste se divide en diferentes niveles de abstracción. Así, el DFD de contexto da la visión más abstracta del sistema: cuales son los agentes externos que interactúan con el sistema y que datos le suministran o solicitan. El DFD de contexto se concreta en el DFD de nivel 1 y, a su vez, cada proceso complejo de dicho DFD se concreta o “explota” en otros DFDs. Así sucesivamente, hasta alcanzar procesos elementales.

Por las razones antes mencionadas, tiene sentido aplicar esta estrategia, que en el diseño y la codificación suele denominarse “refinamiento progresivo”, en la representación de sistemas complejos con cualquier otra notación de análisis. Por ejemplo:

- Un DTE podría descomponerse “explotando” sus estados.
- Un DER podría descomponerse “explotando” sus entidades.

2. ¿Cómo afecta la independencia funcional al mantenimiento de un sistema?

Solución

Los dos criterios básicos para medir la independencia funcional entre los módulos constituyentes de un sistema son la cohesión y el acoplamiento.

La cohesión se refiere a la coherencia del contenido de un módulo. Cuanto mayor sea la cohesión de los módulos, más fácil será localizar los cambios durante el mantenimiento.

Entregue la hoja de lectura óptica con sus datos junto con su examen.

El acoplamiento entre módulos se refiere a la interrelación existente entre ellos. Cuanto menor sea el acoplamiento, menor será la propagación de los cambios durante el mantenimiento (la modificación de un módulo no implicará el cambio de los módulos con los que interactúa).

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

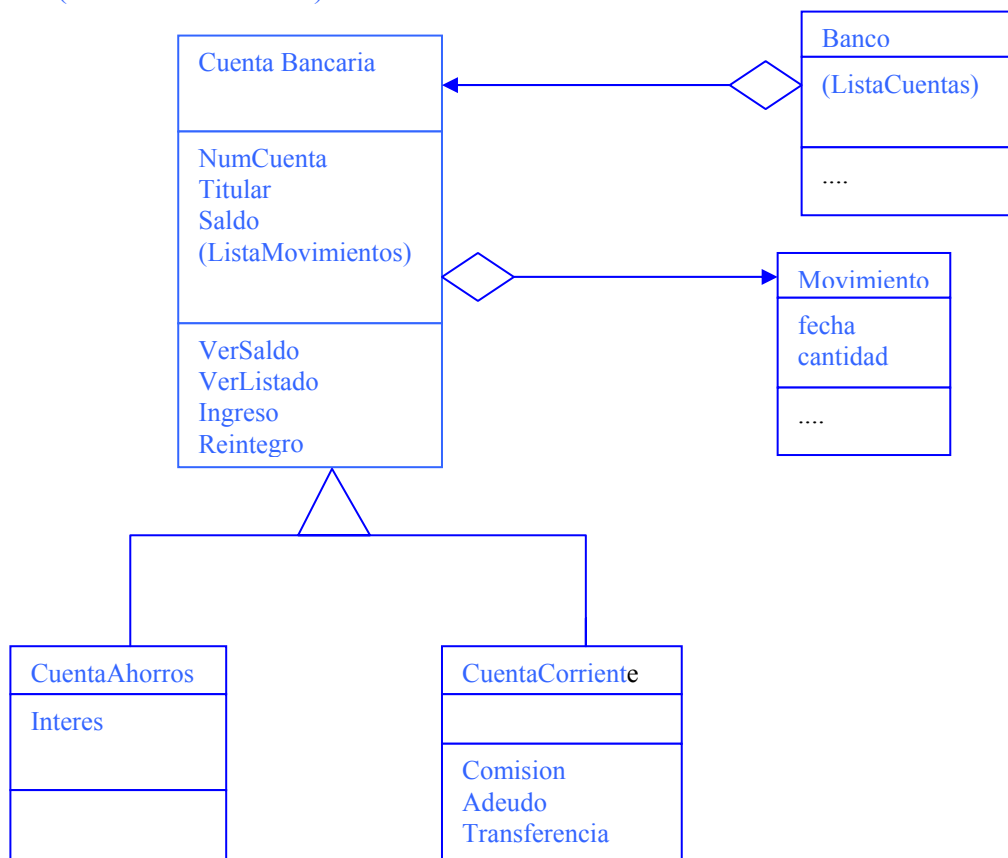
3. Un sistema informático de gestión bancaria opera con dos tipos de cuentas bancarias: la corriente y la de ahorros. Las cuentas disponen de un número que las identifica, tienen asociado un titular, una lista de los movimientos realizados hasta la fecha y almacenan el saldo disponible. Así mismo, las cuentas soportan operaciones como pedir el saldo y hacer ingresos o reintegros. Las cuentas de ahorros producen un interés que se calcula cada mes. Las cuentas corrientes no producen interés sino que reciben la carga de una comisión anual. Dichas cuentas corrientes facilitan diversas operaciones como adeudos domiciliarios y transferencias de dinero. Cada movimiento bancario consta de la fecha, la cantidad y el concepto de que se trata.

Modele la situación anterior utilizando un diseño orientado a objetos. Utilice herencia y composición cuando se pueda.

Solución

CLASES:

- Cuenta de ahorros
- Cuenta corriente
- Cuenta bancaria
- Movimientos
- Banco (Sistema de Gestión)



Entregue la hoja de lectura óptica con sus datos junto con su examen.



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. ¿Qué se entiende por análisis del dominio? ¿Que ventajas produce en el desarrollo del producto software?

SOLUCIÓN:

Por **dominio** entenderemos el campo de aplicación en el que se encuadra el sistema que se construye. En cada campo o dominio existe desde siempre una manera específica de realizar las cosas y una terminología ya acuñada que debe ser respetada y tenida en cuenta. Esto es lo que denominaremos “*realizar un análisis del dominio de la aplicación*”.

Si bien las peculiaridades de cada aplicación hacen que necesariamente deba ser estudiada como un caso único, es importante analizar el dominio de la aplicación para situarla dentro de un entorno mucho más global. Para realizar este análisis es aconsejable estudiar los siguientes aspectos:

- Normativa que afecte al sistema
- Otros sistemas semejantes
- Estudios recientes en el campo de la aplicación
- Bibliografía clásica y actualizada: libros y artículos sobre el tema
- ... etc. ...

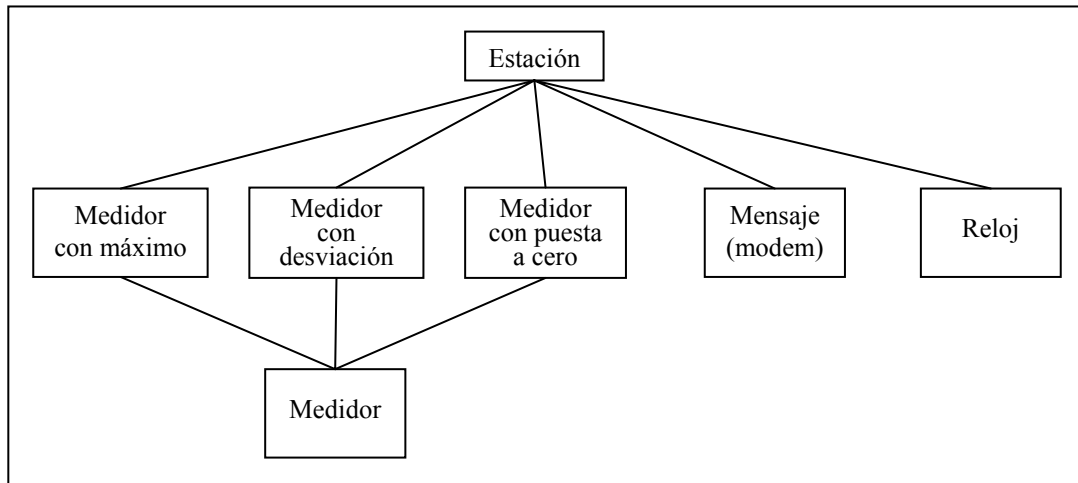
Este estudio facilitará la creación de un modelo más universal. Como ventajas de este enfoque se pueden citar las siguientes:

1. Facilitar la comunicación entre analista y usuario del sistema.
2. Creación de elementos realmente significativos del sistema.
3. Reutilización posterior del software desarrollado.

Apartado 2.1.2.5 del libro de texto, páginas 40 a 42.

Entregue la hoja de lectura óptica con sus datos junto con su examen.

2. Dado el siguiente diagrama de arquitectura (construido con las relaciones de uso entre los módulos), correspondiente al ejemplo de una “estación meteorológica”; desarrolle el diagrama orientado a objetos equivalente.



SOLUCIÓN:

Uno posible es el correspondiente a la figura 4.14, de la página 187, del libro de texto.

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

3. Se desea comprobar la corrección de un programa que calcula el precio de la estancia en un hotel. Los datos de entrada al programa son el día y el mes de la primera noche de estancia y el número de noches. El precio por noche es de:
- 100 € del 7 de enero al 30 de junio y del 1 de septiembre al 23 de diciembre.
 - 140 € del 1 de julio al 31 de agosto.

El hotel permanece cerrado del 24 de diciembre al 6 de enero. La salida que genera el programa es el precio en euros, siempre que se le hayan proporcionado valores correctos a la entrada. Si los datos de entrada no son adecuados, el programa devuelve el texto “datos no válidos”.

Se pide desarrollar un juego de pruebas de error del programa, justificando la elección de los casos escogidos.

SOLUCIÓN:

Puesto que sólo se conoce la especificación entrada-salida del programa, y no su estructura interna, el juego de pruebas será de tipo “caja negra”. Para ello emplearemos, de forma conjunta y complementaria, los métodos de “partición en clases de equivalencia” y “análisis de valores límite”.

A grandes rasgos podemos dividir el espacio de ejecución en tres subespacios:

- temporada baja (2 periodos: 7/1 al 30/6 y 1/9 al 23/12)
- temporada alta (1/7 al 31/8)
- temporada de cierre (24/12 al 7/1)

Y podemos establecer como valores límite las fechas de transición entre estas clases de equivalencia:

- 6/1 y 7/1
- 30/6 y 1/7
- 31/8 y 1/9

Entregue la hoja de lectura óptica con sus datos junto con su examen.

- 23/12 y 24/12

Un juego de casos de prueba podría ser:

CASOS VÁLIDOS		
ENTRADA	COMENTARIO	SALIDA
7/1, 5 noches	Valor límite temporada baja	500 euros (5 x 100)
24/3, 11 noches	1 ^{er} periodo temporada baja	1100 euros (11 x 100)
28/6, 3 noches	Valor límite temporada baja	300 euros (3 x 100)
25/6, 7 noches	Transición baja/alta, valor límite alta	740 euros (6 x 100 + 1 x 140)
1/7, 2 noches	Valor límite temporada alta	280 euros (2 x 140)
12/7, 20 noches	Temporada alta	2800 euros (20 x 140)
20/8, 12 noches	Valor límite temporada alta	1680 euros (12 x 140)
31/8, 2 noches	Transición alta/baja, valores límite	240 euros (100 + 140)
15/11, 6 noches	2º periodo temporada baja	600 euros (6 x 100)
21/12, 3 noches	Valor límite temporada baja	300 euros (3 x 100)
28/6, 70 noches	Transición baja/alta/baja	9480 euros (8 x 100 + 62 x 140)
CASOS NO VÁLIDOS		
ENTRADA	COMENTARIO	SALIDA
3/1, 7 noches	Temporada de cierre	“datos no válidos”
6/1, 2 noches	Valor límite temporada de cierre	“datos no válidos”
19/12, 11 noches	Temporada de cierre	“datos no válidos”
24/12, 4 noches	Valor límite temporada de cierre	“datos no válidos”

Otro caso de estudio, que complica bastante el problema, sería la comprobación de la corrección de la fecha. También se deberían preparar casos de prueba según se trate o no de año bisiesto, etc.

Entregue la hoja de lectura óptica con sus datos junto con su examen.



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 la especificación de requisitos del “Videojuego de las Minas”, indique cuál (o cuáles) de los siguientes requisitos son “de recursos”, funcionales, “de capacidad” o “de operación”:
R.a En todo momento, el jugador estará informado de los segundos transcurridos y de las minas que todavía quedan por marcar del total de las minas ocultas inicialmente.
R.b Para moverse de una casilla a otra de las que la rodean, sólo será necesario pulsar una tecla una sola vez.
R.c Tiempo para situar inicialmente las minas ≤ 1 segundo.

SOLUCIÓN:

R.a: Requisito funcional.

R.b: Requisito de operación.

R.c: Requisito de capacidad.

2. Explique brevemente los conceptos de herencia y polimorfismo empleados en el diseño de software.

SOLUCIÓN:

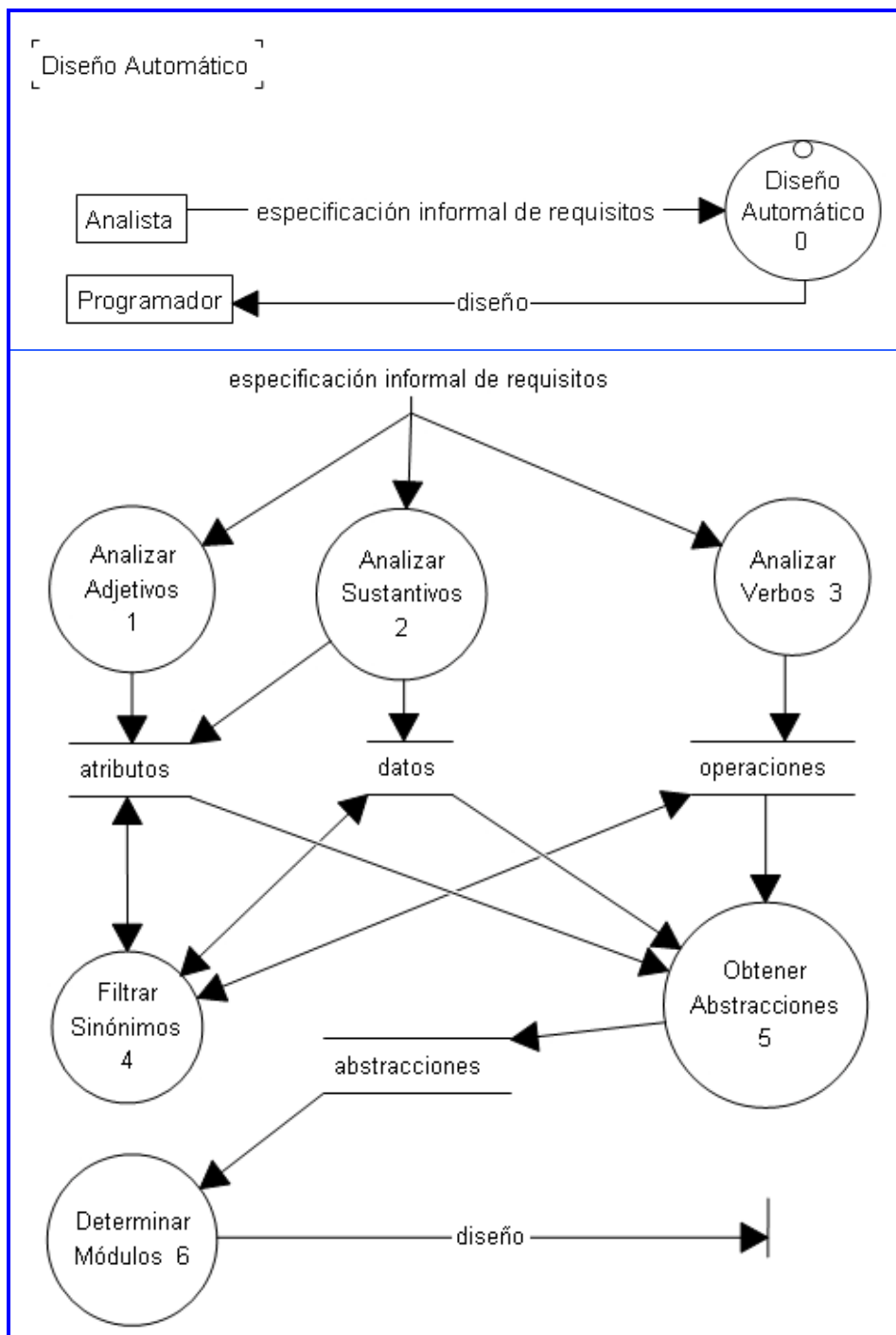
Apartados 3.2.7 y 3.2.8 del libro de texto.

Entregue la hoja de lectura óptica con sus datos junto con su examen.

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

3. Se desea desarrollar un programa informático que, a partir de una especificación informal de requisitos, sea capaz de producir automáticamente un diseño aplicando el método de Abbott. **Modele el programa utilizando un Diagrama de Flujo de Datos.**

SOLUCIÓN:



Entregue la hoja de lectura óptica con sus datos junto con su examen.



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. ¿Cuáles son las tareas o etapas fundamentales en el análisis de requisitos?

SOLUCIÓN:

1. Estudio del sistema en su contexto.
 2. Identificación de necesidades.
 3. Análisis de alternativas. Estudio de viabilidad.
 4. Establecimiento del modelo del sistema.
 5. Elaboración del documento de especificación de requisitos.
 6. Revisión continuada del análisis.
2. ¿Cuáles son las cualidades deseables de una descomposición modular en la fase de diseño de software?

SOLUCIÓN:

Apartado 4.1 del libro de texto.

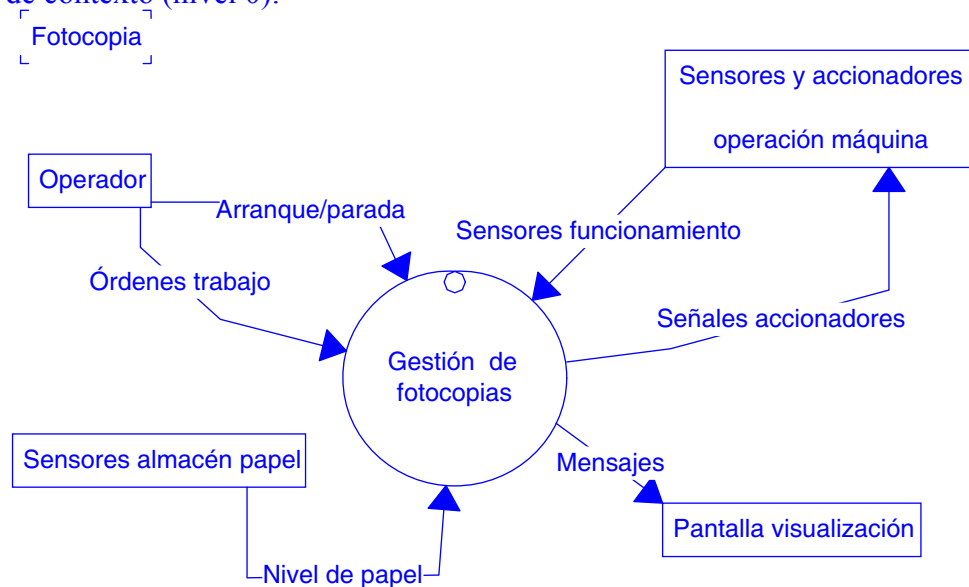
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Desarrolle un modelo de análisis en el que se refleje con claridad y sencillez el comportamiento del software de una fotocopidora. **Represente dicho modelo mediante DFD (hasta el nivel 1 ó 2) Y Diagramas de Transición de Estados.**

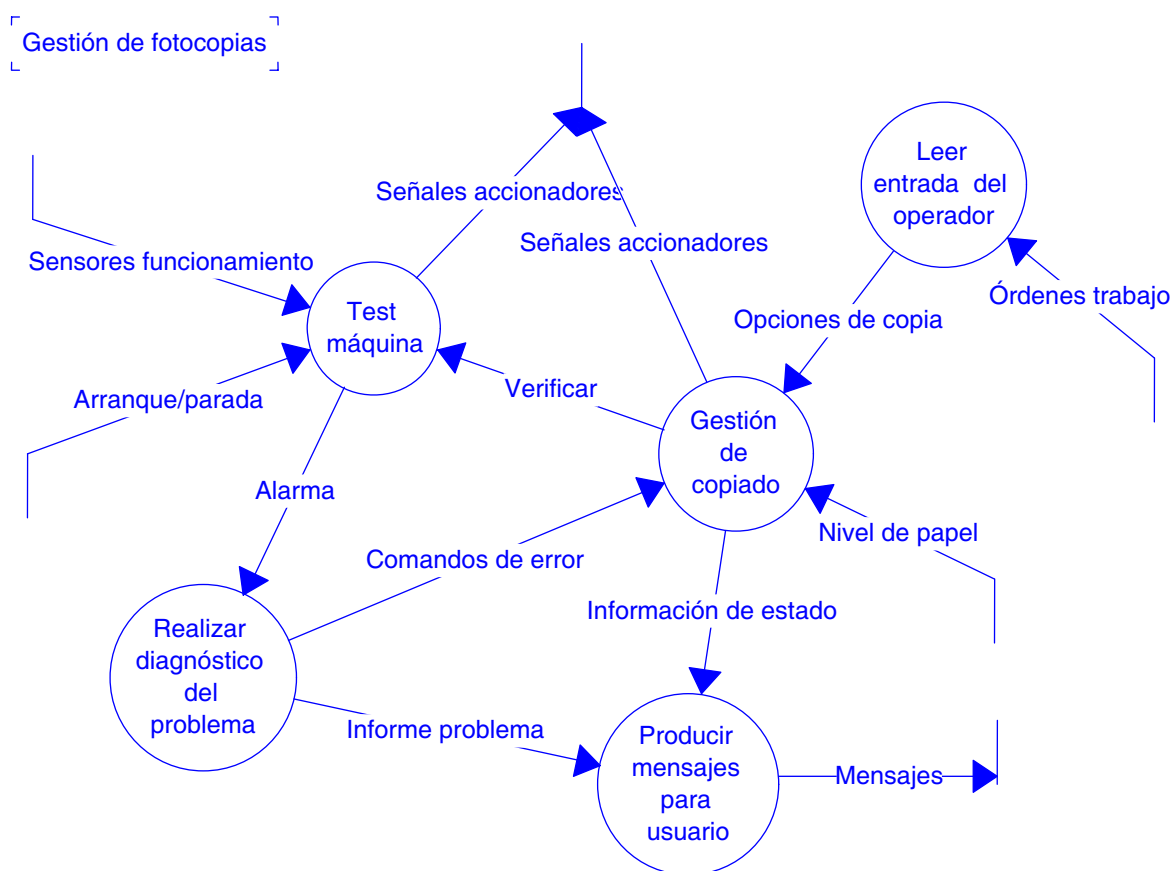
SOLUCIÓN:

Con Diagramas de Flujo de Datos:

Diagrama de contexto (nivel 0):

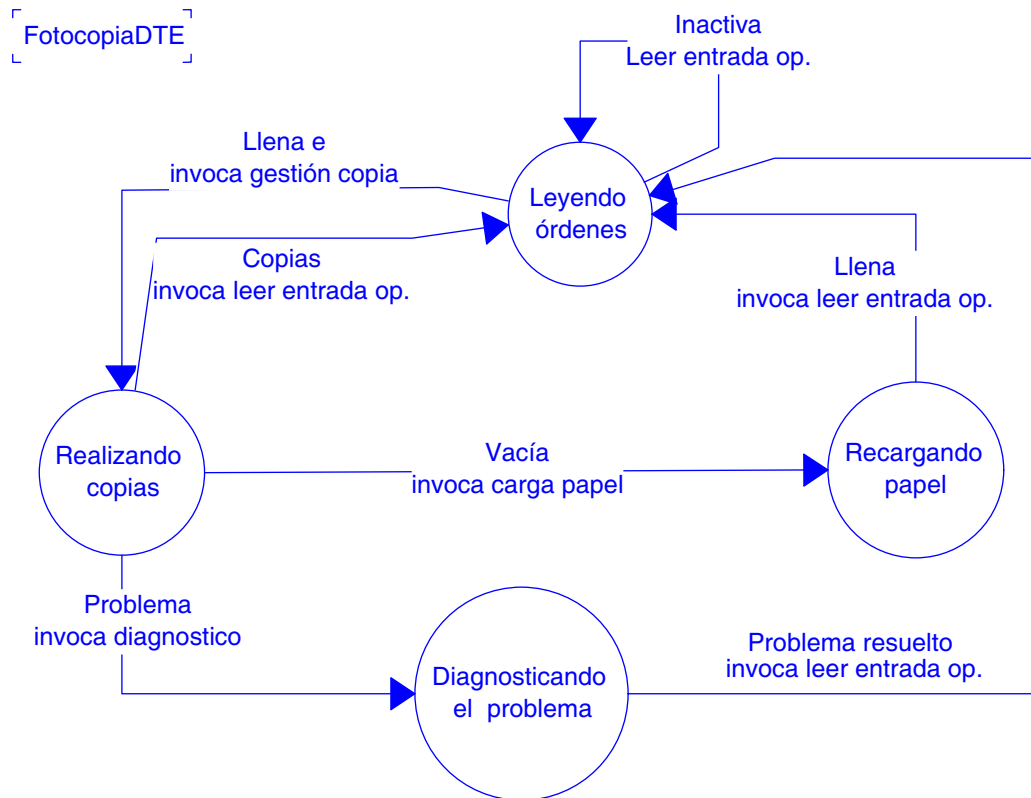


DFD0 (nivel 1):



Entregue la hoja de lectura óptica con sus datos junto con su examen.

Diagrama de Transición de Estados:



Entregue la hoja de lectura óptica con sus datos junto con su examen.



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. ¿Qué se entiende por análisis del dominio? ¿Que ventajas produce en el desarrollo del producto software?

SOLUCIÓN:

Por **dominio** entenderemos el campo de aplicación en el que se encuadra el sistema que se construye. En cada campo o dominio existe desde siempre una manera específica de realizar las cosas y una terminología ya acuñada que debe ser respetada y tenida en cuenta. Esto es lo que denominaremos “*realizar un análisis del dominio de la aplicación*”.

Si bien las peculiaridades de cada aplicación hacen que necesariamente deba ser estudiada como un caso único, es importante analizar el dominio de la aplicación para situarla dentro de un entorno mucho más global. Para realizar este análisis es aconsejable estudiar los siguientes aspectos:

- Normativa que afecte al sistema
- Otros sistemas semejantes
- Estudios recientes en el campo de la aplicación
- Bibliografía clásica y actualizada: libros y artículos sobre el tema
- ... etc. ...

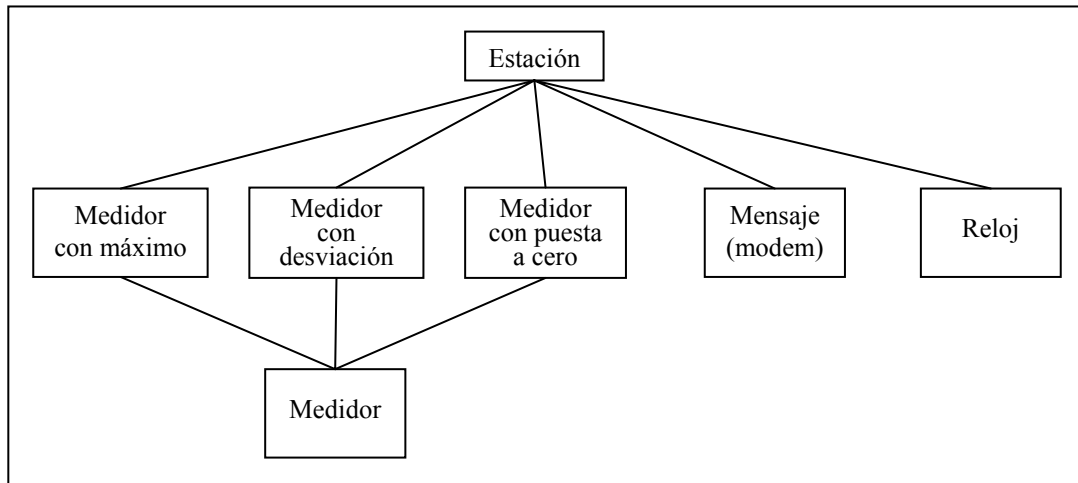
Este estudio facilitará la creación de un modelo más universal. Como ventajas de este enfoque se pueden citar las siguientes:

1. Facilitar la comunicación entre analista y usuario del sistema.
2. Creación de elementos realmente significativos del sistema.
3. Reutilización posterior del software desarrollado.

Apartado 2.1.2.5 del libro de texto, páginas 40 a 42.

Entregue la hoja de lectura óptica con sus datos junto con su examen.

2. Dado el siguiente diagrama de arquitectura (construido con las relaciones de uso entre los módulos), correspondiente al ejemplo de una “estación meteorológica”; desarrolle el diagrama orientado a objetos equivalente.



SOLUCIÓN:

Uno posible es el correspondiente a la figura 4.14, de la página 187, del libro de texto.

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

3. Se desea comprobar la corrección de un programa que calcula el precio de la estancia en un hotel. Los datos de entrada al programa son el día y el mes de la primera noche de estancia y el número de noches. El precio por noche es de:
- 100 € del 7 de enero al 30 de junio y del 1 de septiembre al 23 de diciembre.
 - 140 € del 1 de julio al 31 de agosto.

El hotel permanece cerrado del 24 de diciembre al 6 de enero. La salida que genera el programa es el precio en euros, siempre que se le hayan proporcionado valores correctos a la entrada. Si los datos de entrada no son adecuados, el programa devuelve el texto “datos no válidos”.

Se pide desarrollar un juego de pruebas de error del programa, justificando la elección de los casos escogidos.

SOLUCIÓN:

Puesto que sólo se conoce la especificación entrada-salida del programa, y no su estructura interna, el juego de pruebas será de tipo “caja negra”. Para ello emplearemos, de forma conjunta y complementaria, los métodos de “partición en clases de equivalencia” y “análisis de valores límite”.

A grandes rasgos podemos dividir el espacio de ejecución en tres subespacios:

- temporada baja (2 periodos: 7/1 al 30/6 y 1/9 al 23/12)
- temporada alta (1/7 al 31/8)
- temporada de cierre (24/12 al 7/1)

Y podemos establecer como valores límite las fechas de transición entre estas clases de equivalencia:

- 6/1 y 7/1
- 30/6 y 1/7
- 31/8 y 1/9

Entregue la hoja de lectura óptica con sus datos junto con su examen.

- 23/12 y 24/12

Un juego de casos de prueba podría ser:

CASOS VÁLIDOS		
ENTRADA	COMENTARIO	SALIDA
7/1, 5 noches	Valor límite temporada baja	500 euros (5 x 100)
24/3, 11 noches	1 ^{er} periodo temporada baja	1100 euros (11 x 100)
28/6, 3 noches	Valor límite temporada baja	300 euros (3 x 100)
25/6, 7 noches	Transición baja/alta, valor límite alta	740 euros (6 x 100 + 1 x 140)
1/7, 2 noches	Valor límite temporada alta	280 euros (2 x 140)
12/7, 20 noches	Temporada alta	2800 euros (20 x 140)
20/8, 12 noches	Valor límite temporada alta	1680 euros (12 x 140)
31/8, 2 noches	Transición alta/baja, valores límite	240 euros (100 + 140)
15/11, 6 noches	2º periodo temporada baja	600 euros (6 x 100)
21/12, 3 noches	Valor límite temporada baja	300 euros (3 x 100)
28/6, 70 noches	Transición baja/alta/baja	9480 euros (8 x 100 + 62 x 140)
CASOS NO VÁLIDOS		
ENTRADA	COMENTARIO	SALIDA
3/1, 7 noches	Temporada de cierre	“datos no válidos”
6/1, 2 noches	Valor límite temporada de cierre	“datos no válidos”
19/12, 11 noches	Temporada de cierre	“datos no válidos”
24/12, 4 noches	Valor límite temporada de cierre	“datos no válidos”

Otro caso de estudio, que complica bastante el problema, sería la comprobación de la corrección de la fecha. También se deberían preparar casos de prueba según se trate o no de año bisiesto, etc.

Entregue la hoja de lectura óptica con sus datos junto con su examen.



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. Defina el Ciclo de Vida del Software. Justifique la importancia de este concepto y la necesidad de su utilización.

SOLUCIÓN:

Como aparece en la página 7 del libro de texto: la ingeniería de software amplía la visión del desarrollo de software; a partir de una actividad esencialmente de programación, incorpora un conjunto amplio de actividades adicionales cuya distribución temporal es, precisamente, el ciclo de vida. Así, en la página 10, se define como: el conjunto de actividades involucradas en el proceso de desarrollo de software, así como su organización y distribución temporal, incluyendo el mantenimiento necesario durante su explotación. La justificación de su importancia reside en que la ingeniería de software tiene como objetivo conseguir hacer un seguimiento y control del proceso de desarrollo. Para ello, es necesario disponer de un modelo que identifique qué constituye dicho proceso. Es decir, el ciclo de vida establece una opción para la visibilidad del proceso que se quiere controlar.

2. Indique razonadamente si algún tipo de pruebas de unidades asegura la ausencia de defectos. Resuma brevemente los métodos propuestos en la asignatura para la elaboración de pruebas de caja negra.

SOLUCIÓN:

Ningún tipo de pruebas asegura la ausencia de defectos. Esto sólo puede conseguirse mediante técnicas de verificación formal. Los métodos para la elaboración de pruebas de caja negra están descritos en las páginas 274-279 del libro de texto.

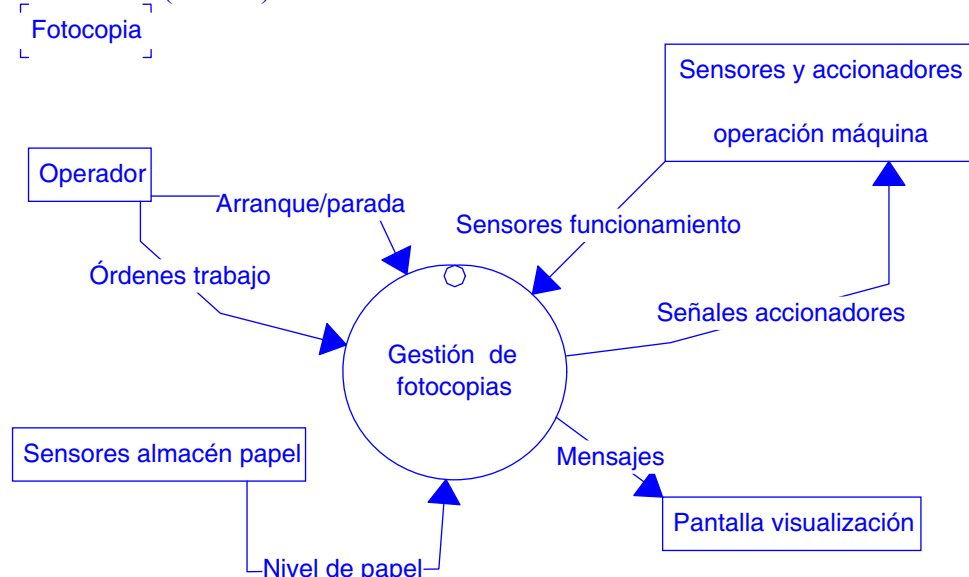
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Desarrolle un modelo de análisis en el que se refleje con claridad y sencillez el comportamiento del software de una fotocopidora (arranque y calibración, programación de tipo de copia, falta de papel, atasco, tóner, etc.) **Represente dicho modelo mediante DFD (hasta el nivel 1 ó 2) Y Diagramas de Transición de Estados.**

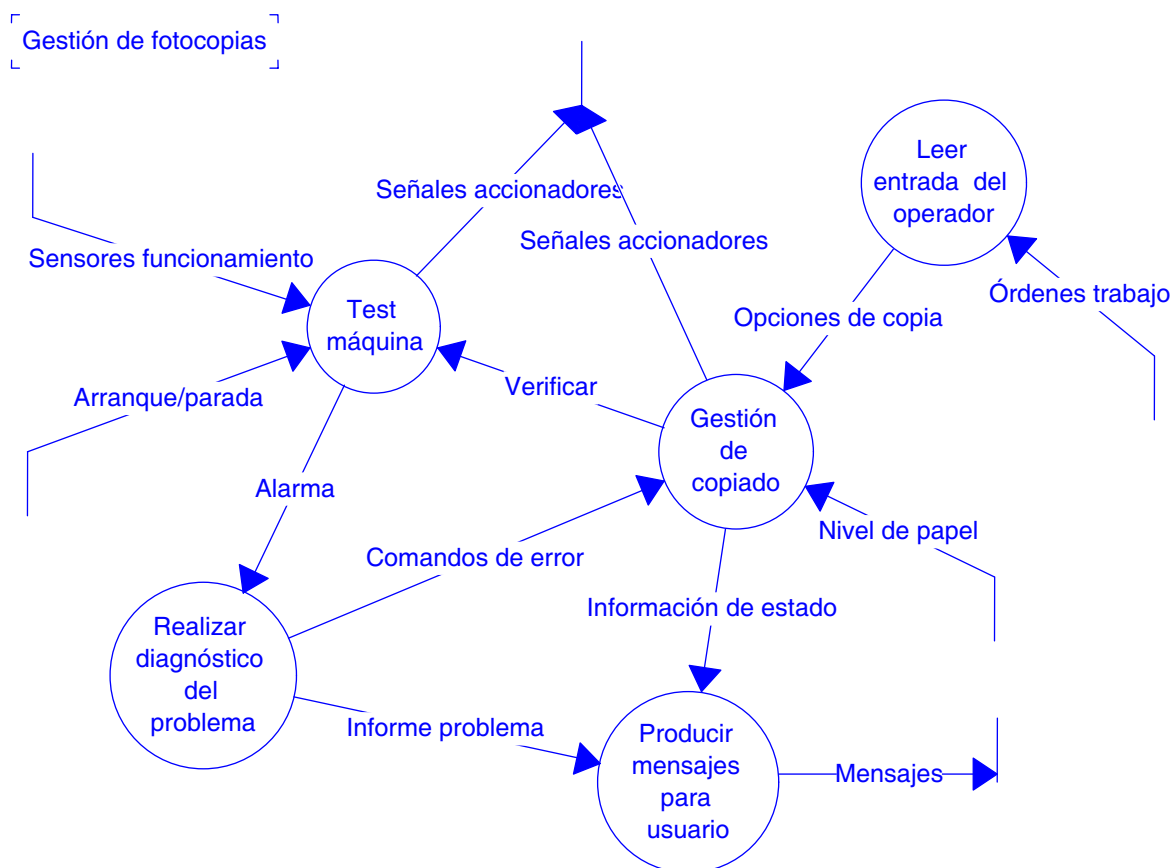
SOLUCIÓN:

Con Diagramas de Flujo de Datos:

Diagrama de contexto (nivel 0):

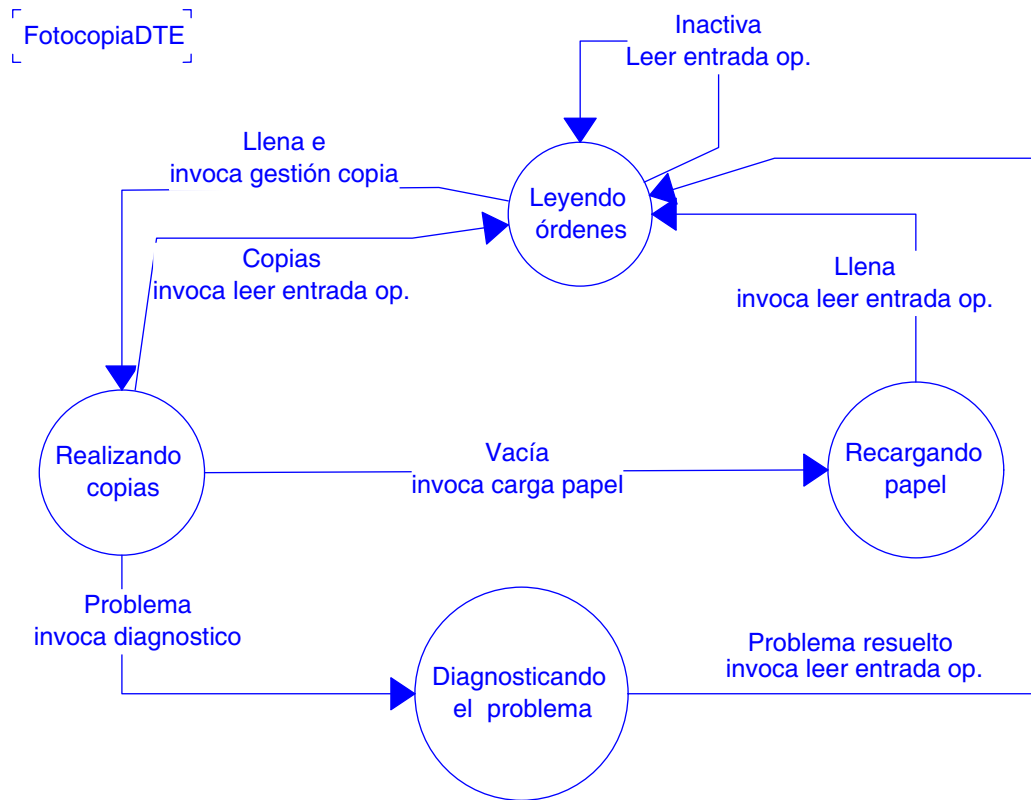


DFD0 (nivel 1):




Entregue la hoja de lectura óptica con sus datos junto con su examen.

Diagrama de Transición de Estados:



Entregue la hoja de lectura óptica con sus datos junto con su examen.

INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	1ª Semana			
				Nacional

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.

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

¡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. Dos aspectos clave que el equipo de analistas debe desarrollar en la fase de especificación de requisitos son: determinar con rapidez las necesidades reales del cliente y proponer unas soluciones funcionales que las satisfagan con eficacia. Piense y explique dos métodos que permitan hacer un seguimiento de estos dos aspectos (por ejemplo con medidas de algún parámetro del proceso en la fase de análisis).

Solución

Para el primer aspecto, determinar con rapidez y agilidad las necesidades del cliente, una medida podría ser el número de cambios en los requisitos que se identifica durante la fase de análisis y elicitación. Un número elevado podría indicar un problema en la comunicación con el cliente o un estudio pobre del dominio. En cuanto a las propuestas de soluciones que satisfagan las necesidades del cliente, se podría contemplar revisando cuantos cambios se producen en los requisitos o en el modelo, en el resto de las fases del desarrollo. Un valor elevado indicaría una comprensión pobre del sistema que se desarrolla, es decir, un análisis incompleto o defectuoso que obligaría a frecuentes 'vueltas atrás' y replanteamientos.

Nota: Esta pregunta se ha calificado con el siguiente algoritmo:

La calificación es el máximo entre uno (1) y la puntuación obtenida en la corrección '*normal*' de la respuesta (puntuación máxima de 2'5).

2. En que medida incorpora o facilita, el Diseño Orientado a Objetos, la aplicación de los conceptos de '*abstracción*', '*modularidad*', '*ocultación*', '*herencia*' y '*polimorfismo*'.

Solución

- A. En cuanto a la abstracción, el concepto de clase nos permite manejar información como elementos individuales abstrayéndonos de sus posible composición interna y atendiendo sólo a su comportamiento.
- B. La modularidad: los módulos de nuestro diseño básicamente son las clases. Se relacionan entre ellas por medio del modelo de clases y el de relaciones de uso.
- C. Ocultación, propiedad por la cual conseguimos que nuestros módulos oculten sus estructura interna, también se consigue utilizando Clases, pues, su implementación se realiza dentro de la propia clase y es totalmente transparente cuando se utilizan éstas en la aplicación.

- D. Herencia, es un mecanismo propio del diseño Orientado a Objetos, que nos permite, de entrada, reutilizar código.
- E. Polimorfismo. La herencia, a parte de reutilizar código, tiene como principal característica el permitir polimorfismo: de anulación o diferido.

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

3. Se ha codificado en Modula-2 el siguiente subprograma que distingue, por el tamaño de sus lados si un triángulo es isósceles (dos lados iguales), equilátero (todos los lados iguales) o escaleno (ningún lado igual).

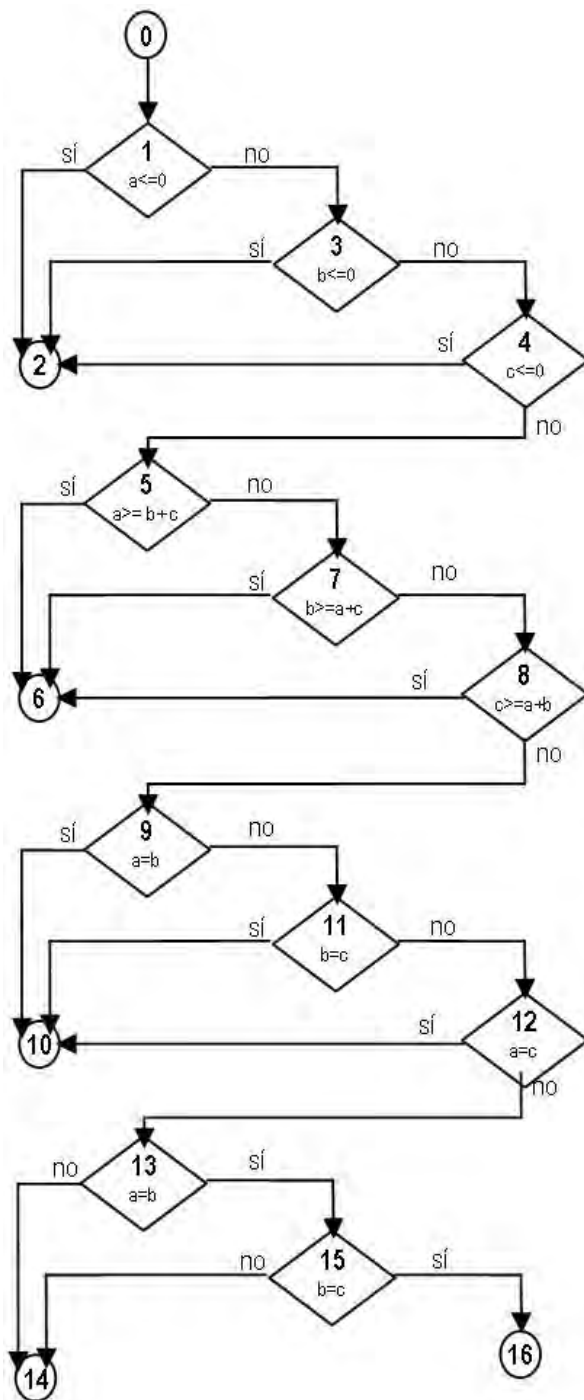
```
PROCEDURE TipoDeTriangulo(a, b, c: INTEGER);
BEGIN
  IF (a<=0) OR (b<=0) OR (c<=0) THEN
    WriteString("Error: alguno de los parámetros no es un número natural");
  ELSIF (a>=b+c) OR (b>=a+c) OR (c>=a+b) THEN
    WriteString("Error: no es posible cerrar el triángulo");
  ELSIF (a=b) OR (b=c) OR (a=c) THEN
    WriteString("El triángulo es isósceles");
  ELSIF (a=b) AND (b=c) THEN
    WriteString("El triángulo es equilátero");
  ELSE
    WriteString("El triángulo es escaleno");
  END;
END TipoDeTriangulo;
```

Verifique el subprograma con pruebas de caja transparente, realizando el cubrimiento lógico correspondiente.

Solución

Para la verificación del subprograma se elaborará un conjunto de casos de prueba que consigan que se transite por todos los posibles caminos de ejecución y que pongan en juego todos los elementos del código.

En primer lugar, el código del subprograma se transformará en el siguiente diagrama de flujo, donde cada rombo representa un predicado lógico simple:



② Parámetros erróneos: valores no naturales

⑥ Parámetros erróneos: imposible cerrar el triángulo

⑩ Triángulo isósceles

⑭ Triángulo escaleno

⑮ Triángulo equilátero

A continuación, se calculará el nº de caminos básicos para recorrer todas las líneas de flujo del diagrama al menos una vez:


Nº de predicados = 11

Nº máximo de caminos = Nº de predicados + 1 = 12

A continuación se determinarán los caminos y los casos de prueba que forzarán su recorrido.

Como indica la siguiente figura, es imposible escribir un juego de prueba para los caminos 0-1-3-4-5-7-8-9-12-13-15-14 y 0-1-3-4-5-7-8-9-12-13-15-16. Es decir, **el subprograma es incorrecto** por que contiene dos caminos que no pueden recorrerse. Concretamente, la imposibilidad de transitar por el camino 0-1-3-4-5-7-8-9-12-13-15-16 hace que el subprograma clasifique erróneamente los triángulos equiláteros como isósceles.

	Camino	Resultado esperado	Juego de prueba
1	0-1-2	Parámetros erróneos: valores no naturales	$a=0, b=1, c=1$
2	0-1-3-2		$a=1, b=-3, c=1$
3	0-1-3-4-2		$a=3, b=1, c=0$
4	0-1-3-4-5-6	Parámetros erróneos: imposible cerrar el triángulo	$a=5, b=1, c=1$
5	0-1-3-4-5-7-6		$a=2, b=4, c=2$
6	0-1-3-4-5-7-8-6		$a=3, b=1, c=5$
7	0-1-3-4-5-7-8-9-10	Triángulo isósceles	$a=2, b=2, c=1$
8	0-1-3-4-5-7-8-9-11-10		$a=2, b=3, c=3$
9	0-1-3-4-5-7-8-9-12-10		$a=4, b=3, c=4$
10	0-1-3-4-5-7-8-9-12-13-14	Triángulo escaleno	$a=3, b=4, c=5$
11	0-1-3-4-5-7-8-9-12-13-15-14		Es imposible crear un juego de prueba
12	0-1-3-4-5-7-8-9-12-13-15-16	Triángulo equilátero	

INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Original			
		Unión Europea		

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.

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

¡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. 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. ¿Cuáles son las distintas estrategias de integración de los módulos de un producto software? Explíquelas brevemente.

Solución

(Pág. 285 y siguientes).

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

3. Una clínica dental quiere informatizar su sistema de información. De ella recibimos el siguiente texto escrito:

“Deseamos almacenar registros de los pacientes, en los que figuren sus datos personales (nombre, dirección, teléfono, antecedentes médicos, etc.), las dolencias detectadas en sus visitas a nuestra clínica y los tratamientos realizados.

Disponemos de un equipo médico encargado de realizar los diagnósticos y llevar a cabo los tratamientos a los pacientes.

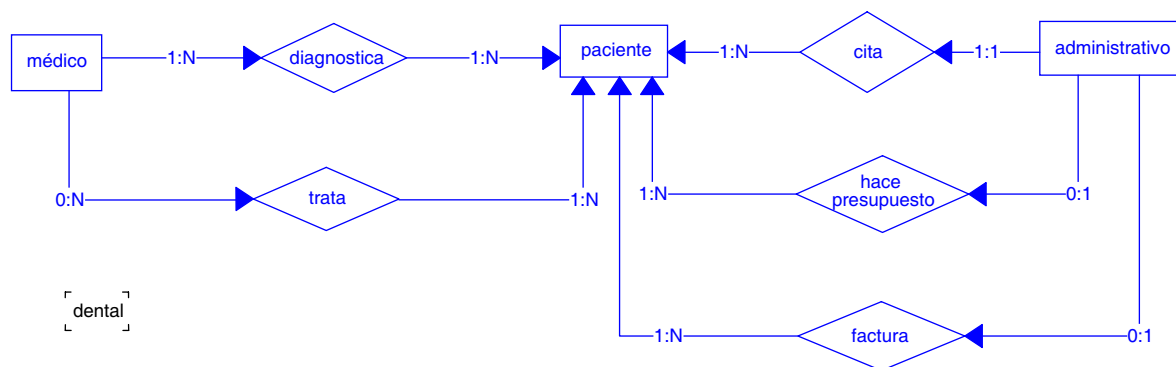
El personal administrativo se encarga de citar a los pacientes, realizar presupuestos de los tratamientos recomendados por los médicos y facturar los tratamientos realizados a los pacientes”

Se pide:

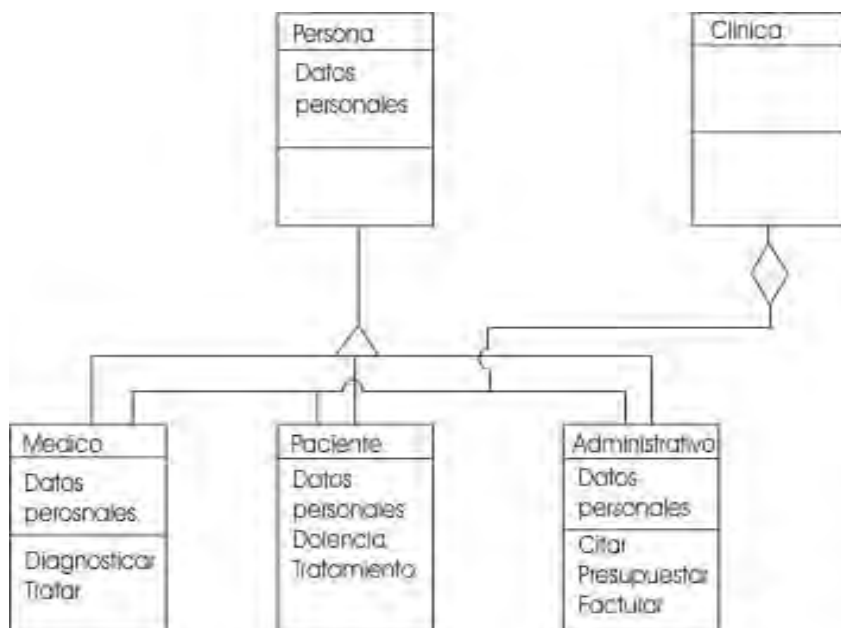
- **Realice un análisis de los datos que va a manejar el sistema mediante el modelo entidad-relación.**
- **Establezca la estructura modular del sistema mediante la técnica de diseño orientado a objetos.**

Solución

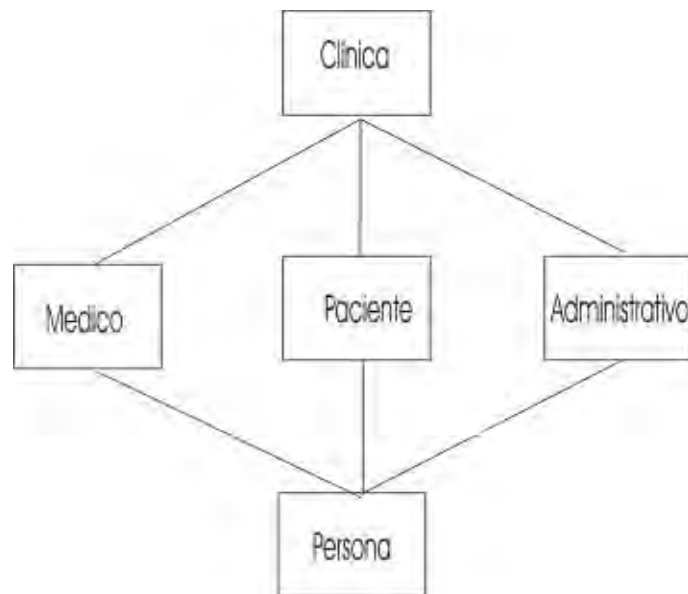
Diagrama entidad-relación:




Modelo de objetos:



Para obtener la estructura modular asignamos un módulo a cada clase de objetos. Las relaciones de herencia se traducen a relaciones de uso y las subclases utilizan el código de su superclase:



INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Original			
				C. Penitenciarios

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.

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

¡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. Razone qué criterios deben guiar la elección de un modelo de ciclo de vida.

Solución

Algunos de los criterios que deben guiar la selección de un modelo de ciclo de vida son los siguientes:

- Volatilidad de los requisitos. ¿Qué prestaciones ofrecen el modelo para reaccionar ante cambios en los requisitos?
- Incertidumbre asociada al proyecto debido a la falta de experiencia en proyectos similares, la dificultad de comprender el requisito, el uso de tecnologías novedosas...
- Facilidad que el modelo da a los gestores para controlar el progreso de los sistemas.
- Rapidez con que el usuario dispone de parte o de todo el sistema.

El modelo de ciclo de vida en cascada es el más antiguo y el más ampliamente utilizado. Debido a su simplicidad, el control del progreso de los sistemas es muy sencillo. Además, no carga el ciclo de vida con actividades que ralentizan la entrega del sistema total. Sin embargo, dada la dificultad de volver atrás, no responde eficazmente a cambios en los requisitos ni maneja de manera apropiada la incertidumbre.

El modelo evolutivo corrige la necesidad de una secuencia no lineal de pasos de desarrollo, gestiona eficazmente la volatilidad de los requisitos y la incertidumbre, y permite la rápida entrega parcial del sistema ("por fascículos").

Respecto al modelo evolutivo, en el modelo en espiral:

- Existe un reconocimiento explícito de las diferentes alternativas para alcanzar los objetivos de un proyecto.
- La identificación de riesgos asociados con cada una de las alternativas y las diferentes maneras de resolverlos son el centro del modelo. Con el modelo evolutivo es habitual dejar las partes más difíciles para el final y empezar con las más fáciles y de menor riesgo, obteniendo así la ilusión de un gran avance.

- La división de los proyectos en ciclos, cada uno con un acuerdo final de cada ciclo, implica que existe un acuerdo para los cambios que hay que realizar o para terminar el proyecto, en función de lo que se ha aprendido desde el inicio del proyecto.

Respecto al modelo de ciclo de vida en cascada, en los modelos evolutivos y en espiral se complica el control del progreso de los sistemas y se introducen nuevas actividades que pueden ralentizar la entrega total de sistemas sencillos.

2. Describa las estructuras de programa recomendadas por la metodología de *programación estructurada*. Explique la equivalencia existente, empleada en la *metodología de Jackson* (Diseño Dirigido por los Datos), entre las estructuras de programa y las estructuras de datos.

Solución

(Pág, 128, 162, 244 y 253).

Las estructuras de programa recomendadas por la metodología de programación estructurada son la **secuencia**, la **selección** y la **iteración**. Cualquier lenguaje de programación imperativo proporciona sentencias para facilitar la programación estructurada:

- secuencia: escritura consecutiva de sentencias
- selección: construcciones *if – then – else, case – of, etc.*
- Iteración: construcciones *while – do, repeat – until, for – to – do, loop y exit.*

La *metodología de Jackson* es un procedimiento de diseño sistemático de software a partir de las estructuras de los datos de entrada y salida del sistema. La equivalencia que existe entre las estructuras de programa anteriores y las estructuras de datos es la siguiente:

- secuencia – tupla (estructura de datos compuesta de elementos heterogéneos; la mayoría de los lenguajes disponen de la estructura registro o *record* para su implementación).
- selección – unión (también se usa el tipo registro para definir el esquema unión, que es una agrupación de elementos posibles de tipos diferentes; en el caso de Pascal y Modula-2 correspondería al registro con campos variantes).
- iteración – formación (colección de elementos del mismo tipo; la estructura que proporcionan los lenguajes se denomina formación, vector o *array*).

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

3. Actualmente, las bicicletas de montaña están compuestas, entre otras piezas, por un manillar, un sistema de cambio, un sillín, una horquilla de suspensión, dos pedales, una cadena y un cuadro. La cadena está formada por un conjunto de eslabones y el cuadro, dependiendo de si la bicicleta es de “doble suspensión” o no, dispone de un amortiguador.

En la asignatura se estudian dos notaciones para modelar datos, que en ocasiones pueden considerarse equivalentes: los diccionarios de datos y los diagramas entidad relación. **Modele el enunciado anterior utilizando ambas notaciones. En el diccionario de datos omita el campo “Utilidad” y las descripciones de los componentes de la bicicleta (límitese a cumplimentar los campos “Nombre” y “Estructura”).**

Solución

a) Diccionario de datos:

Nombre: Bicicleta de montaña

Estructura: Manillar + Cambio + Sillín + Horquilla de suspensión + {Pedal}² + Cadena + [Cuadro rígido | Cuadro con suspensión]

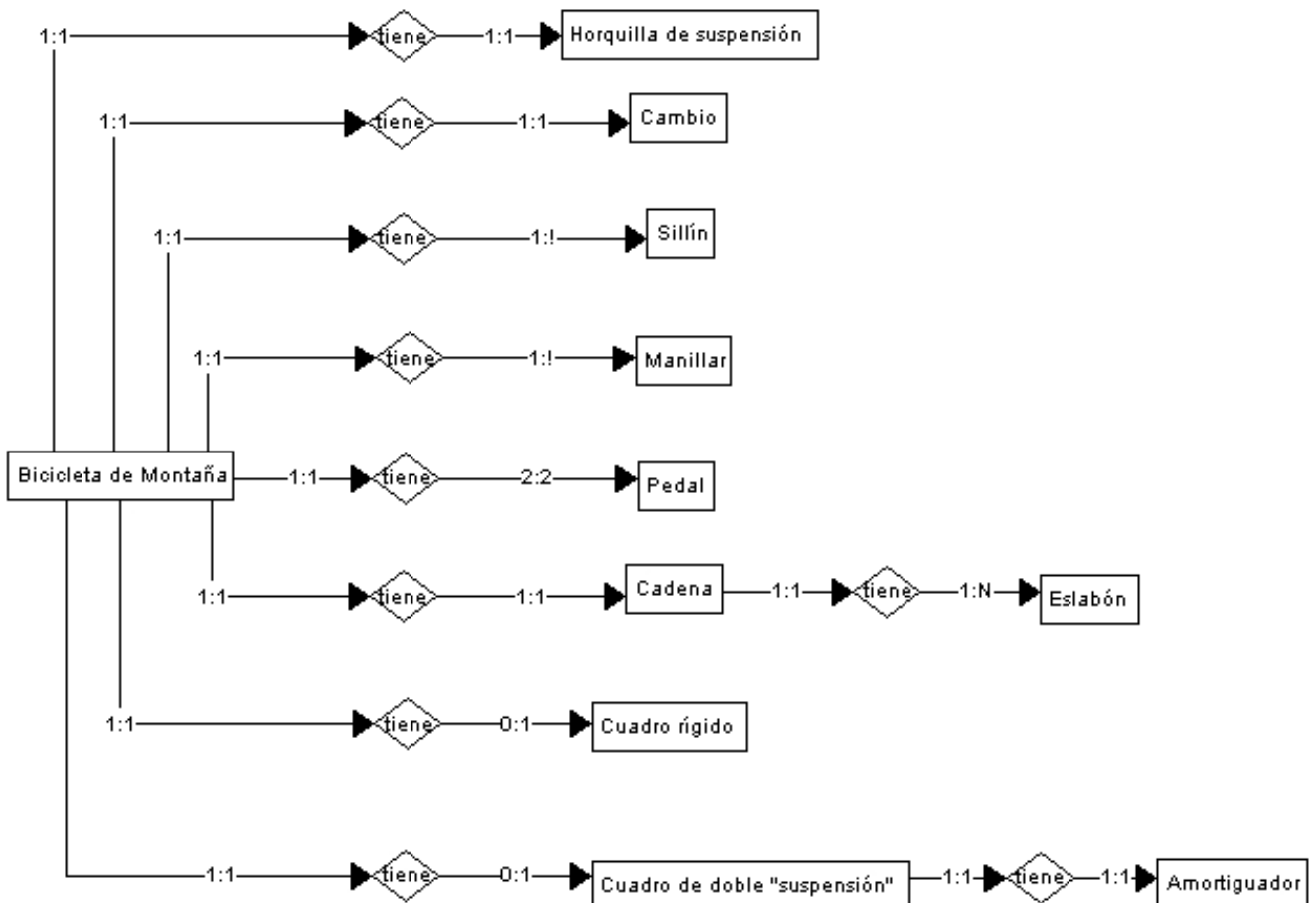
Nombre: Cadena


Estructura: {Eslabón}

Nombre: Cuadro con suspensión

Estructura: Amortiguador

b) Diagrama entidad relación:



INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Septiembre 2007 Original			
				Nacional y UE

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

NECESITO EL EJERCICIO MANUSCRITO POR EL ALUMNO 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. Frente a la fabricación '*intuitiva*' o '*artesanal*' de un producto software ¿qué ventajas tiene el uso de las técnicas de ingeniería del producto software, vistas en la asignatura (en cuanto al uso de un ciclo de vida, el análisis, el diseño, la codificación, cómo se hacen las pruebas, la integración, etc.), respecto al producto final obtenido y el proceso de su desarrollo?

Solución

- A. Ciclo de vida. Disponer de un modelo de referencia para el comportamiento del proceso de desarrollo que permite la visibilidad, el seguimiento y control de las actividades implicadas en la fabricación del producto.
- B. El Análisis. Permite alcanzar la comprensión de las necesidades del cliente, la situación, el objetivo, el comportamiento y la funcionalidad de lo que tendrá que ser la aplicación cuando se construya. El conjunto de actividades de esta fase establecen la definición del producto. Por tanto, la otra ventaja, además de la comprensión, es que se establecen, de manera explícita, un conjunto de especificaciones, más o menos formales, **imprescindibles** para el diseño, la codificación, validación y aceptación del producto.
- C. En el diseño se define un modelo teórico del funcionamiento del sistema. El análisis de estas especificaciones de funcionamiento permite deducir: defectos en la funcionalidad del sistema o de alguno de sus componentes; el grado de independencia entre módulos, su comprensibilidad y transportabilidad y, por tanto, se pueden estimar los posibles problemas de mantenibilidad, reutilización y codificación. El refinamiento del diseño reduce los problemas anteriores y facilita la verificación, la integración y las pruebas.
- D. En la codificación se realiza la construcción '*física*' del producto. Cuando se hace teniendo en cuenta las pautas de la Ingeniería del Producto Software y basándose en la realización correcta de las fases anteriores, se garantizan mejores resultados en cuanto a los plazos de entrega, la mayoría de los aspectos de calidad del producto; detección, control y eliminación de defectos, facilidad de integración y reutilización futura del código.

- E. Un buen programa de pruebas permite detectar el máximo de defectos de parte del producto o del sistema completo. Esta detección, sobre todo si es precoz, permite disminuir los plazos de entrega, facilita la eliminación de defectos (calidad) y ahorra mucho trabajo. Las pruebas se pueden y deben realizar durante todas las fases del ciclo de vida. De esta manera se posibilita la visibilidad y el control del desarrollo del producto y se permite que los replanteamientos sean tempranos y eficaces. En cuanto a las pruebas del código, del funcionamiento de los componentes y del sistema; la identificación de los defectos y de su origen depende, en gran medida, de la calidad del diseño realizado y de que la ejecución de la codificación haya sido ordenada y controlada, bien documentada y se hayan realizado las ‘*buenas prácticas*’ recomendadas. El proceso de corrección, cambios y mantenimiento se facilitará tanto más cuanto se incorporen estas técnicas en el desarrollo.
 - F. Integración y pruebas del sistema. Las ventajas mencionadas anteriormente para el diseño, la codificación y pruebas modulares, se extienden y aplican a la facilidad y éxito de la integración; tanto en el caso de grandes aplicaciones como en las de pequeño tamaño que requieran una integración mínima. En cualquier caso, los problemas que se puedan detectar serán de origen más fácilmente identificable con un diseño claro, una descomposición modular altamente independiente, un código ‘*limpio*’ y bien documentado que si este es farragoso o si el diseño tiene dependencias fuertes.
 - G. La gestión de la configuración y, en concreto, el establecimiento y mantenimiento de la línea base, permite que todas las actividades del desarrollo se realicen en direcciones que son compatibles entre sí, aunando los esfuerzos en el sentido de la finalización del producto. Las especificaciones de la línea base se van estableciendo paulatinamente a lo largo de todo el ciclo de vida, en una sucesión de distintas líneas base de las que sólo una está vigente (congelada) en cada instante. Como las especificaciones de la línea base garantizan que los trabajos que se realicen son compatibles entre sí y con lo anteriormente desarrollado, se evita la dispersión de esfuerzos y el trabajo que supone la adecuación de una parte del desarrollo para que sea compatible con el resto.
2. Resuma en qué consiste el principio de ocultación y qué ventajas se derivan de su aplicación.

Solución

El principio de ocultación y sus ventajas se resumen en las páginas 113 y 114 de J. A. Cerrada Somolinos, et al. Introducción a la Ingeniería del Software. Ed. Ramón Areces, 2000. A continuación, se da una explicación complementaria a dicho resumen:

¿Qué es el principio de ocultación? Es un principio que promueve esconder todos los detalles que sean irrelevantes para utilizar un determinado artefacto.

Los lenguajes de programación ofrecen diversos medios para seguir este principio. Por ejemplo, los detalles de implementación de los subprogramas suelen ocultarse mediante cabeceras, que muestran exclusivamente cuales son los parámetros de entrada y salida de los subprogramas.

¿Qué ventajas se derivan de su aplicación? El principio de ocultación es “de propósito general”, ya que su aplicación va más allá del mero desarrollo de software. Por ejemplo, la electrónica de los equipos de música se esconde dentro de una caja que ofrece una escueta interfaz con los botones ON, OFF, PLAY... ¿Qué ocurriría si no fuera así y para escuchar un disco tuviéramos que configurar cables, transistores...?

1. La utilización del equipo exigiría años de estudio para conocer la electrónica del equipo.
2. Si el equipo se estropea y adquirimos un nuevo modelo, habría que volver a invertir tiempo en aprender qué cables, transistores... hay que configurar. Es decir, al estar “acoplados” a la tecnología del equipo, si ésta cambia, nos vemos forzados a cambiar nosotros también.
3. La prueba de los equipos se encarecería, ya que, además de expertos en acústica, forzosamente deberán participar expertos en electrónica.

Centrándonos en el desarrollo de software, el principio de ocultación facilita:

1. La reutilización de componentes desarrollados por terceros.

2. El mantenimiento del código que reutiliza componentes desarrollados por terceros. Mientras los componentes mantengan su interfaz, la evolución de los componentes no impondrá cambios en el código que los reutiliza.
3. La realización de pruebas de caja negra.

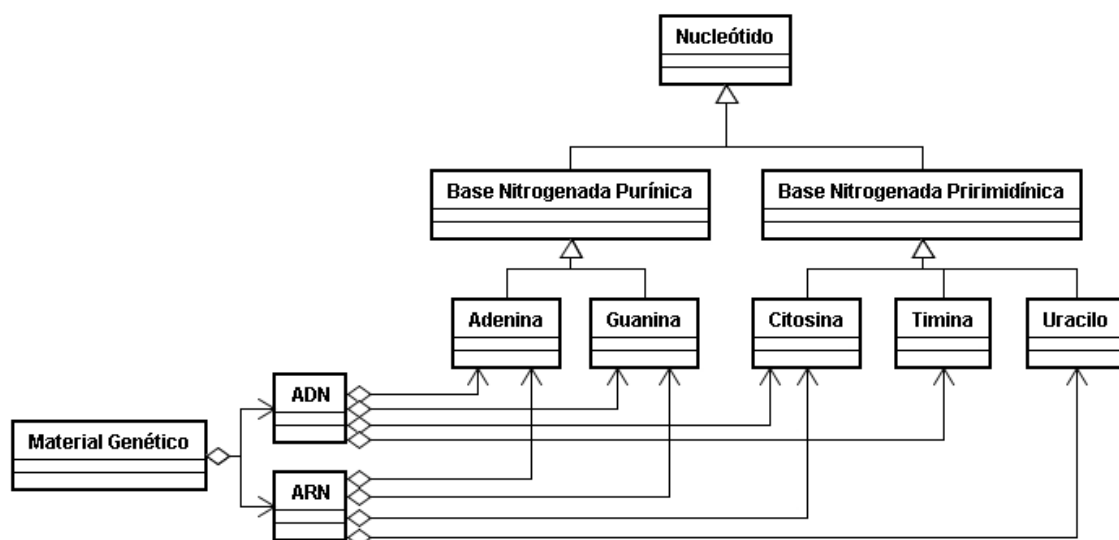
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Los principales componentes del material genético de los organismos son el ADN (Ácido DesoxirriboNucleico) y el ARN (Ácido RiboNucleico). El ADN está formado por cuatro tipos de nucleótidos, dos con bases nitrogenadas purínicas: la adenina y la guanina, y dos con bases nitrogenadas pirimidínicas: la citosina y la timina. El ARN está compuesto por los mismos tipos de nucleótidos, salvo que en lugar de timina contiene uracilo.

Modele el enunciado anterior mediante un Diagrama de Objetos.

Solución

Del análisis del enunciado se desprenden las relaciones de agregación y composición: **componentes** del 'material genético'; 'ADN' y 'ARN' **formado por...** Por otro lado, cualquiera de los compuestos que forman el materia genético **son** 'bases nitrogenadas', de un tipo u otro, y **son** 'nucleótidos'; de donde se deducen las relaciones de herencia.



INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Sept - 2007 Reserva			
				Nacional y UE

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

NECESITO EL EJERCICIO MANUSCRITO POR EL ALUMNO 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. Resuma gráficamente los modelos de ciclo de vida en V, con prototipos evolutivos y en espiral.

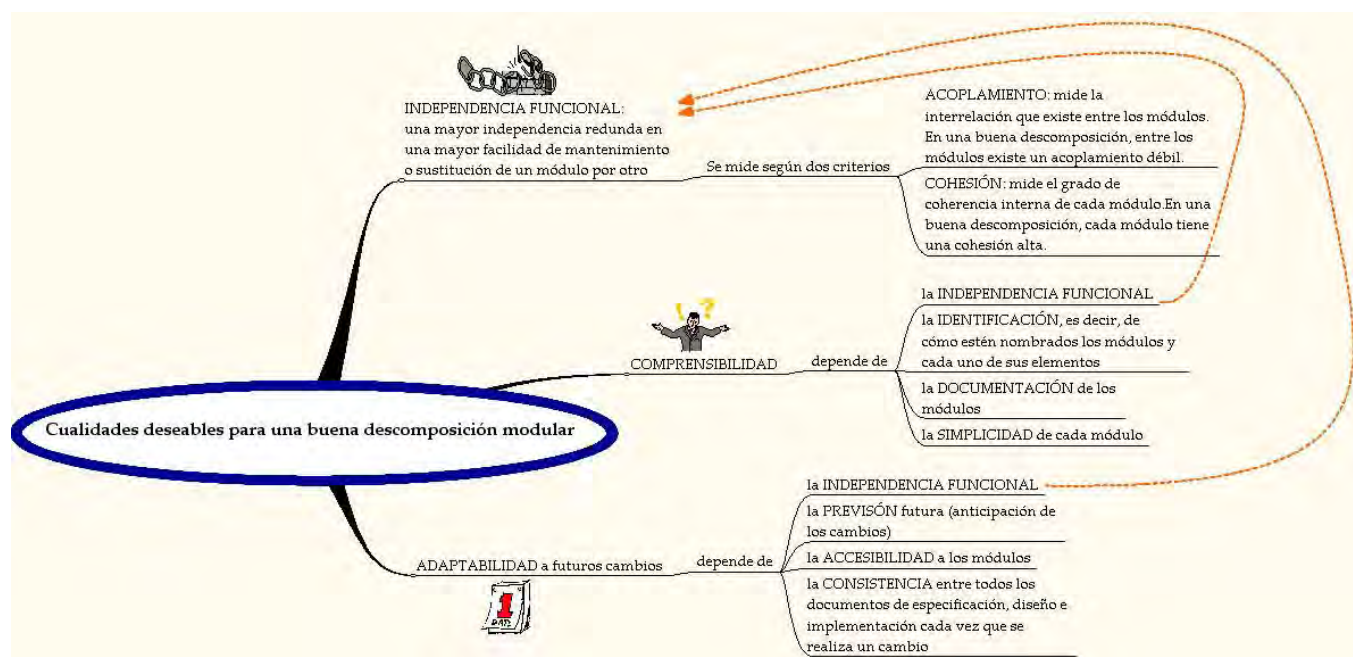
Solución

Figuras 1.3, 1.6 y 1.7 de J. A. Cerrada Somolinos, et al. *Introducción a la Ingeniería del Software*. Ed. Ramón Areces, 2000.

2. Resuma qué cualidades debe poseer una buena descomposición modular.

Solución

Las cualidades que debe poseer una buena descomposición modular se indican en el apartado 4.1 de J. A. Cerrada Somolinos, et al. *Introducción a la Ingeniería del Software*. Ed. Ramón Areces, 2000. A continuación, se incluye un esquema que resume dichas cualidades.



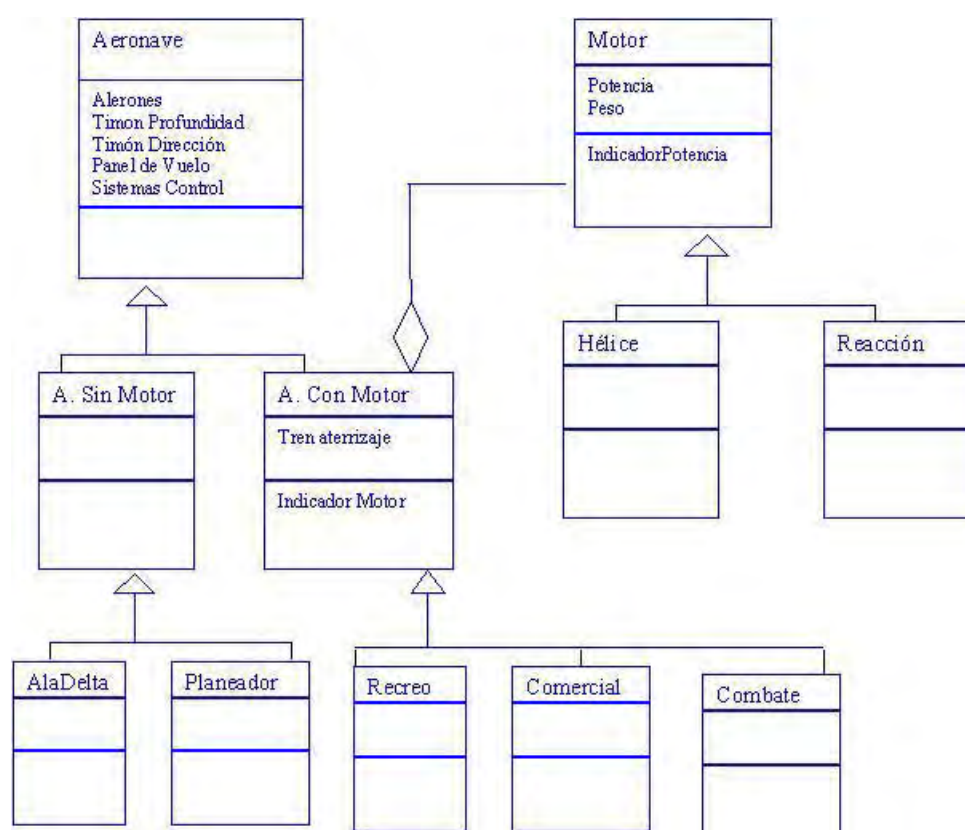
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. En un juego de simulación aérea, el usuario puede elegir entre distintos tipos de aeronaves: ala-delta, planeador, ultraligero, avioneta de recreo, avión de combate, etc. Todas las aeronaves se rigen por el mismo principio básico de navegación aérea, sin embargo, cada uno ofrece además su propio comportamiento dependiendo si se trata de una aeronave con motor o sin él, si éste es de hélice o a reacción o, incluso, del número de motores de que dispone.

Diseñe un modelo de objetos que represente los distintos modelos de aeronave. Utilice herencia siempre que pueda y justifique sus pasos en la construcción del diseño. Intente añadir funcionalidad a las clases en forma de algunos métodos que se le ocurran.

Solución

Para poder identificar las clases que tendrá nuestro diseño, emplearemos las directivas de Abbot.



Dado que las características básicas de vuelo son comunes a todos los aviones, y atendiendo a criterios de abstracción y coherencia, parece natural diseñar una clase padre llamada *Aeronave* que recoja estas características. A partir de ella irán heredando las distintas clases de aeronaves que dispongamos. Antes, sin embargo, hacemos la distinción entre aeronaves de motor y sin motor ya que su comportamiento e instrumentación serán distintas; estas dos clases serán las *A. Sin Motor* y *A. Con Motor* desde las cuales heredaran todas las distintas clases de aeronaves que dispondrá nuestro juego.

Por otro lado, dado que el motor juega una parte importante en la navegación de las aeronaves, se ha diseñado una clase *Motor* que recogerá la funcionalidad básica que tendrán los dos tipos de motores: los de hélice y los de reacción.

INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Sept - 2007 Original			
				C. Penitenciarios

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

NECESITO EL EJERCICIO MANUSCRITO POR EL ALUMNO 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)

- En el texto base de la asignatura, al presentar los distintos modelos de ciclo de vida, se ilustran los conceptos de 'verificación' y 'validación' solamente al hablar del ciclo de vida en V (ciclo de vida que es idéntico al de cascada –clásico como él- y, lo único que aporta es el ámbito en el que se desarrollan las actividades de cada fase).
 - Diferencie estos dos conceptos entre sí.
 - Razone si la verificación y la validación son actividades sólo del ciclo de vida en V o, por el contrario, dichas actividades tienen que aparecer en cualquier modelo de ciclo de vida.

Solución

- Verificación es la comprobación de que cada parte, módulo, componente o elemento codificado funciona y se comporta correctamente según lo especificado en el diseño.

Por el contrario, la validación es la comprobación de que el sistema, ya integrado, cumple las expectativas del cliente que se hayan reflejado en la documentación del análisis. Dicho de otra manera, en la validación se comprueba que las especificaciones que se han acordado con el cliente y que se han reflejado en la documentación del análisis, se ejecutan satisfactoriamente.

- Por el propio contenido de los dos conceptos: comprobar que cada parte del código se comporta como se esperaba y constatar que se cumplen las necesidades acordadas con el cliente; estas actividades forman parte, necesariamente, de una u otra fase de todos los ciclos de vida. Es inaceptable que no se compruebe si el código funciona o si el producto desarrollado es el que quería el cliente.

2. Explique brevemente las dos estrategias de pruebas de módulos estudiadas en la asignatura.

Solución

Para evitar el caos que supondría el tener que hacer una prueba global única a todo el sistema, antes de la integración, se deberán hacer pruebas a cada módulo de diseño conforme avance la codificación.

Existen dos estrategias fundamentales de prueba de módulos o unidades:

- Pruebas de caja negra
- Pruebas de caja transparente

Las **pruebas de caja negra** se basan exclusivamente en la comprobación de las especificaciones de entrada y salida del módulo. Son pruebas que ignoran la estructura interna del módulo, ya que solo lo 'ven' como una caja negra donde a una determinada entrada deberá corresponder una salida perfectamente determinada.

El objetivo de las pruebas no es 'probar' de forma exhaustiva todas las distintas posibles entradas, sino descubrir errores o incorrecciones del módulo. Por eso, la elección de un buen conjunto de casos de prueba, conseguirá detectar el máximo número de errores con el menor número de pruebas. Estos casos de prueba se pueden escoger mediante distintos métodos, entre ellos podemos destacar:

- Partición de clases
- Análisis de valores medios
- Comparación de versiones
- Empleando la intuición y experiencia

Las **pruebas de caja transparente** tienen como objetivo 'determinar' la salida de un módulo ante una entrada pero teniendo en cuenta la estructura interna del módulo, esto es, se trata de conseguir que la ejecución del módulo transite por todos los posibles caminos de ejecución y ponga en juego todos los elementos del código. Para conseguir casos de pruebas eficaces podemos aplicar las siguientes estrategias:

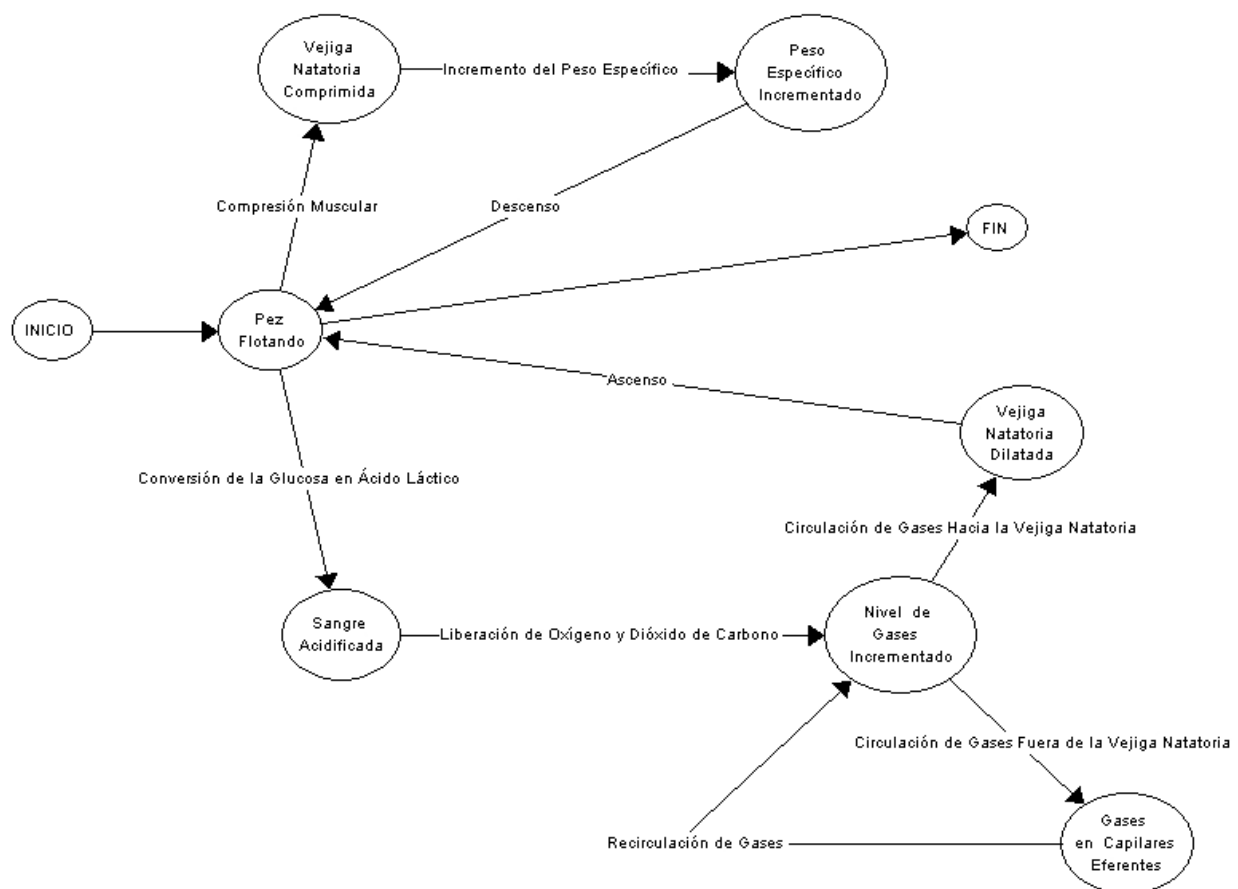
- Cubrimiento lógico
- Pruebas de bucle
- Empleo de la intuición

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

3. La vejiga natatoria es un órgano de flotación que poseen muchos peces. Se trata de una bolsa de paredes flexibles, llena de gas, situada dorsalmente por debajo de la columna vertebral y por encima del tubo digestivo. Controla la flotabilidad mediante un complejo sistema de intercambio gaseoso con la sangre, y permite al pez ascender o descender en el agua. Cuando expulsa gases por compresión muscular, aumenta el peso específico, facilitando el descenso en el agua. Si se llena de gases, favorece el ascenso hacia la superficie. El proceso de secreción gaseosa se basa en la acidificación de la sangre producida al convertirse la glucosa en ácido láctico. Con ello se libera en la sangre oxígeno de la oxihemoglobina y dióxido de carbono del bicarbonato sódico, que se difunden en la vejiga natatoria. Los gases que no llegan inmediatamente hasta la vejiga natatoria pasan a los capilares eferentes y circulan de nuevo.

Modele el movimiento ascendente y descendente de los peces, enunciado en el párrafo anterior, mediante un Diagrama de Transición de Estados.

Solución



INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Sept - 2007 Original			
				Soto del Real

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

NECESITO EL EJERCICIO MANUSCRITO POR EL ALUMNO 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. Describa cinco factores de calidad del software.

Solución

Páginas 27 y 28 de J. A. Cerrada Somolinos, et al. Introducción a la Ingeniería del Software. Ed. Ramón Areces, 2000.

2. ¿Para qué sirven las pruebas de sistema?

Solución

El principal objetivo de las pruebas es conseguir que el programa funcione incorrectamente, esto es, las pruebas deben ser capaces de encontrar posibles errores de software.

En particular se llaman pruebas de sistema a aquellas que se realizan sobre todo el sistema software una vez terminado la integración del mismo. Es este escenario, es cuando se realizan pruebas para evaluar criterios de calidad. Se pueden agrupar en las siguientes clases según el objetivo perseguido:

- Pruebas de recuperación.
- Pruebas de seguridad.
- Pruebas de resistencia.
- Pruebas de rendimiento.

Por último, para ver que el sistema es realmente útil al usuario final, se realizan las pruebas alfa y beta.

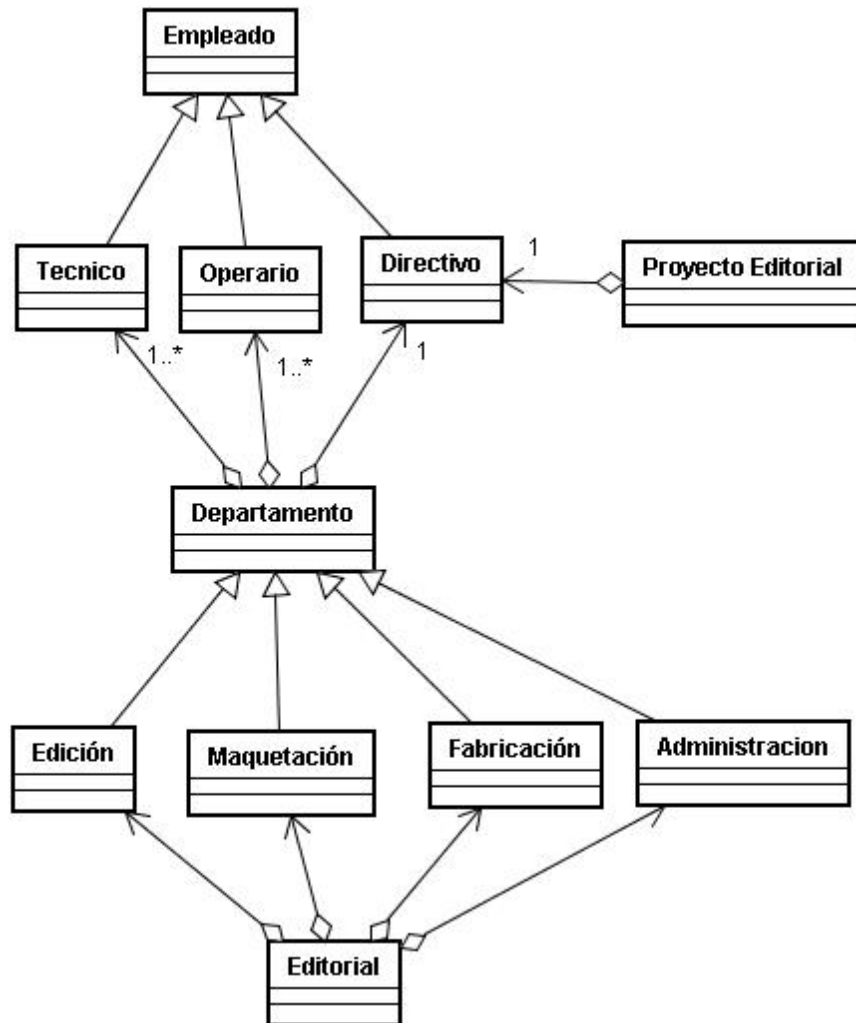
Véase la sección 5.9 del libro de referencia (pág. 289 y 290).


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

3. Una compañía editorial está dividida en 4 departamentos: edición, maquetación, fabricación y administración. Los trabajadores se dividen entre: directivos, técnicos y operarios. Un proyecto editorial implica a todos los departamentos y tiene un directivo como responsable. Los demás empleados están adscritos, al menos, a un proyecto. El jefe de un departamento es un directivo.

Modele el sistema anterior utilizando orientación a objetos.

Solución



INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN			
	Modelo		
	Junio 2008		Ámbito
			1ª SEMANA

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

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. Explique qué se entiende por “pseudocódigo” y para qué se emplea principalmente.

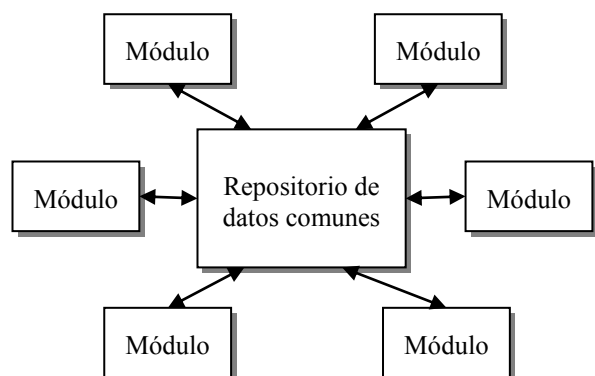
Solución

El pseudocódigo es una notación empleada para la especificación de los **requisitos funcionales** de un sistema. Se basa en las estructuras básicas de los lenguajes de programación estructurada (secuencia, selección e iteración), pero sin los aspectos de declaración de tipos, variables, constantes o subprogramas. No se trata de una notación con reglas estrictas sino que cada analista tendrá su propio estilo a la hora de utilizar el pseudocódigo. Esta notación también sirve para realizar el diseño del sistema, en cuyo caso se estará llegando a un mayor nivel de detalle sobre los requisitos funcionales, sin llegar a la codificación, pero sí detallando el algoritmo que se empleará. Existen pseudocódigos para utilizar en la fase de diseño, que se denominan PDLs (lenguajes de descripción de programas).

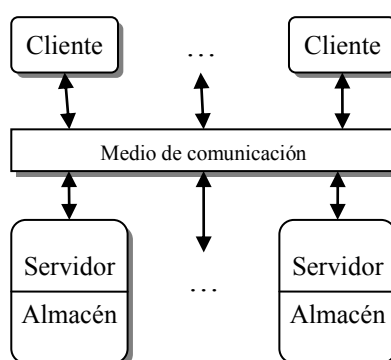
2. Una de las decisiones importantes que hay que tomar al diseñar la arquitectura de un sistema consiste en definir el «*estilo arquitectónico*». Según Garlan y Shaw, un estilo arquitectónico es un patrón de organización de un sistema que incluye estos tres importantes aspectos del diseño:

- 1) La estructura más adecuada (distribución y organización de los módulos y sus dependencias).
- 2) Cómo se descomponen los subsistemas en sus componentes.
- 3) Cómo se controla la ejecución de los subsistemas.

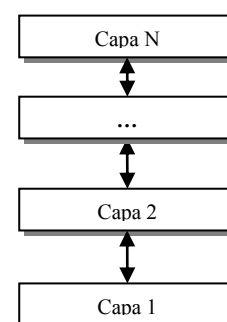
En relación al primer aspecto, tres estilos organizativos ampliamente utilizados son los siguientes:



a) Modelo de repositorio



b) Modelo Cliente-Servidor



c) Modelo por capas

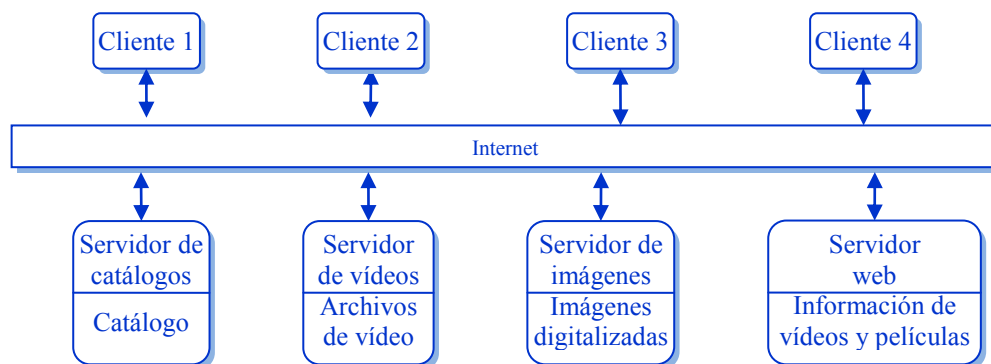
Ahora, estudie este caso con atención:

«Un sistema multiusuario basado en Web, proporciona una biblioteca de películas y fotografías. En él, varios servidores gestionan y visualizan diferentes tipos de dispositivos. Las secuencias de vídeo necesitan ser transmitidas rápidamente y en sincronía, aunque con una resolución relativamente baja. Éstas se pueden comprimir en un almacén para que el servidor de vídeo pueda gestionar la compresión y descompresión de vídeo en diferentes formatos. Sin embargo, las fotografías deben mantenerse con una resolución alta, por lo que es adecuado gestionarlas en un servidor separado. Un gestor de catálogos debe ser capaz de manejar una gran variedad de peticiones y proporcionar enlaces al sistema de información Web, que incluye datos sobre las películas de vídeo y las fotografías, y un sistema de comercio electrónico que soporta la venta de las fotografías y de las películas. Cada programa que usa estos servicios no es más que una interfaz de usuario integrada con estos servicios y construida usando un navegador Web.»

Seleccione, entre los tres de más arriba, el modelo de estructura (organización) de la arquitectura más adecuado para este sistema. Represente el diagrama seleccionado completando las cajas con los elementos que se enuncian para este sistema. Argumente porqué ha tomado esta decisión y qué ventajas tiene.

Solución

La ventaja más importante del modelo cliente-servidor es que es una arquitectura distribuida. Se puede hacer un uso efectivo de los sistemas en red con muchos procesadores distribuidos. Es fácil añadir un nuevo servidor e integrarlo con el resto del sistema o actualizar los servidores de forma transparente sin afectar al resto del sistema. Puede no haber un único modelo de datos compartido entre los servidores y, por tanto, los subsistemas pueden organizar sus datos de formas diferentes. Esto quiere decir que se puede optimizar el rendimiento de cada servidor estableciendo en él un modelo de datos específico.



b) Modelo Cliente-Servidor

Sin embargo, un inconveniente podría ser que, para obtener mayores beneficios de la integración de un nuevo servidor, fuera necesario realizar cambios en los clientes o en los servidores existentes. Si se usara una representación de los datos basada en XML, sería relativamente simple realizar las conversiones de un esquema a otro –en el supuesto de que se hiciera una modificación en algún elemento y el sistema no compartiera un modelo de datos único–. Desgraciadamente, XML es una forma de representar los datos poco eficiente y, su uso, podría acarrear problemas en el rendimiento cuya facilidad de optimización se mencionaba como ventaja.

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

3. El sistema Mail Secure sirve para enviar correos electrónicos firmados y/o cifrados. El funcionamiento básico (simplificado) es el siguiente:

Cada usuario tiene dos claves: una privada, solo accesible por él, y otra clave pública que es compartida por el resto de los usuarios.

Veamos con un ejemplo cómo Alicia envía un mensaje firmado y luego cifrado (oculto) a Bernardo, su amigo.

- 1) Alicia pasa su mensaje por una función hash que lo comprime a modo de resumen.
- 2) Este mensaje hash o resumen se codifica con la clave privada de Alicia. El mensaje resultante se conoce como firma digital.
- 3) El mensaje original queda firmado al unirlo con su firma.
- 4) Alicia puede enviar el mensaje firmado (original + firma) a Bernardo.
- 5) Pero si Alicia quisiera además cifrar (codificar) su mensaje antes de enviarlo, utilizaría la clave pública de Bernardo para cifrar el mensaje firmado obteniendo así un mensaje firmado y cifrado (encriptado).
- 6) Ahora Alicia envía a Bernardo su mensaje firmado y cifrado (encriptado) a salvo de cualquier mirada y además llevando la marca de autenticidad (su firma).

Veamos ahora como procede Bernardo para descifrar el mensaje y comprobar su autoría.

- 1) Cuando Bernardo recibe el mensaje cifrado y firmado de Alicia pasa a descifrarlo utilizando su clave privada, obteniendo el mensaje original de Alicia y su firma.
- 2) Para comprobar que es Alicia quien lo ha firmado, descifra la firma con la clave pública de Alicia. Por otra parte, aplica la función hash al mensaje descifrado de Alicia. Obteniendo así dos mensajes resúmenes.
- 3) Si coinciden, se puede garantizar que la firma es de Alicia, si no, se rechaza el mensaje.

Modele el sistema anterior mediante dos DFD's: uno para la parte emisora (Alicia) y otro para la parte receptora (Bernardo) para el caso que se quiera enviar un mensaje cifrado y firmado.

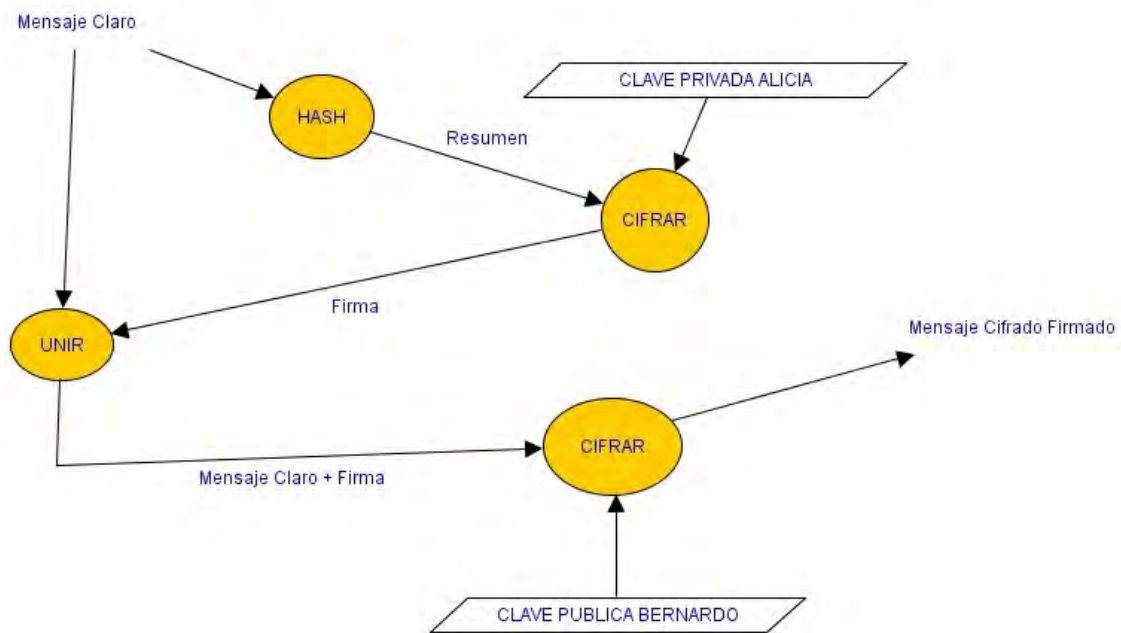
Nota. Codificar, cifrar o encriptar es aplicar una transformación a un texto utilizando un algoritmo que necesita un parámetro llamado clave. La finalidad es ocultar el texto original obteniendo otro texto ilegible. Descifrar o desencriptar es la operación inversa.

Solución

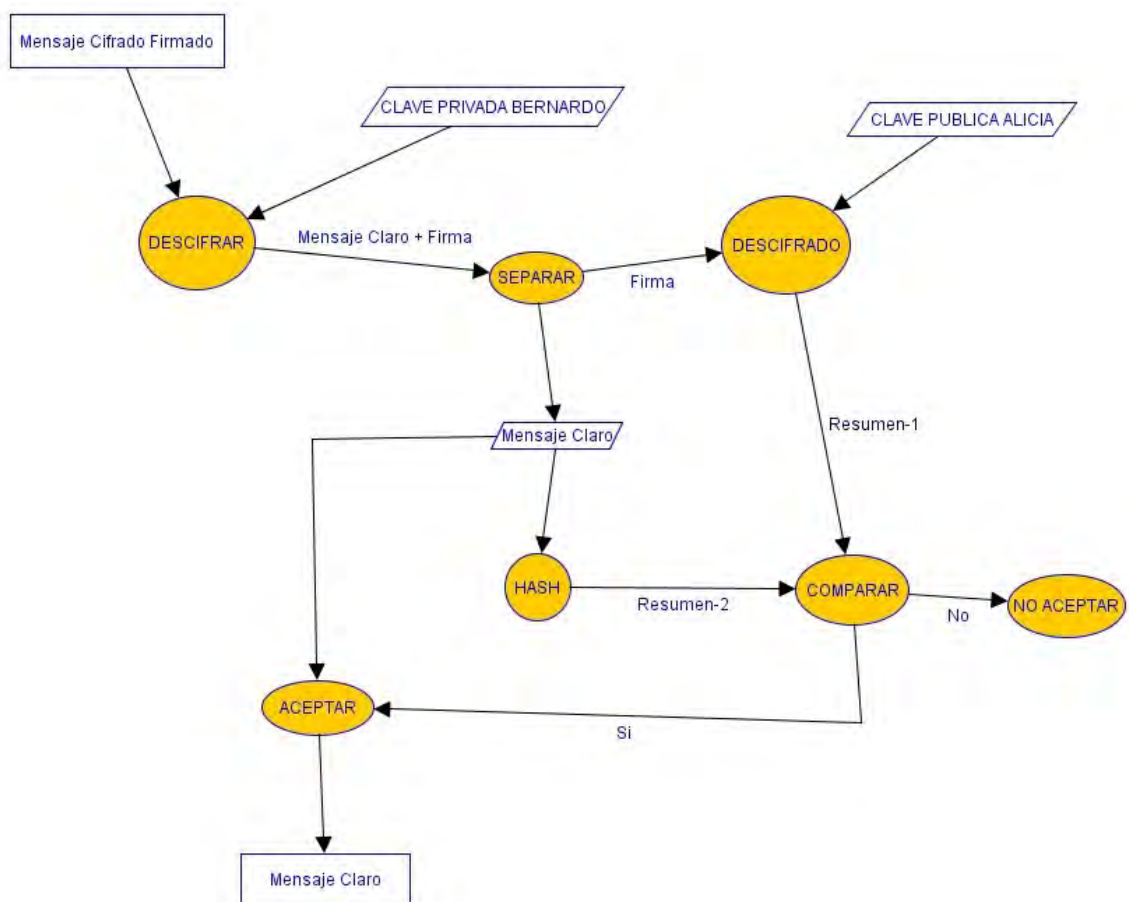
Emisor (Alicia):




DFD de contexto. Nivel 0



Receptor (Bernardo):



INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Modelo			
	Junio 2008			
				Ámbito
				UE 2ª SEMANA

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

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 el “análisis del dominio” del sistema software que se desarrolla y qué ventajas tiene su realización?

Solución

Se trata del estudio del entorno en el que se enmarcará el sistema que se desarrolla, en el cual existirá una manera de hacer las cosas y una terminología que deberá ser tenida en cuenta. Esta es una actividad que se debe contemplar en la fase de análisis del proceso de desarrollo. Al estudiar el contexto del sistema, el analista podrá alcanzar una **comprensión más clara y precisa** de lo que se necesita desarrollar; fundamental para que las actividades tengan éxito. Además, permite tener una **comunicación fluida** con aquellos interlocutores con los que deberá interactuar para llevar a cabo la especificación del sistema. Si se ignora este contexto, la solución estará excesivamente particularizada al caso concreto que pide un cliente, por lo que la solución desarrollada no servirá para otras empresas. Por tanto, el conocimiento de la problemática del entorno servirá al analista para situar la aplicación en un contexto más global, lo que facilitará la **reutilización** posterior del software desarrollado.

2. Explique cómo se lleva a término, y con qué tipo de pruebas, la fase de validación del sistema.

Solución

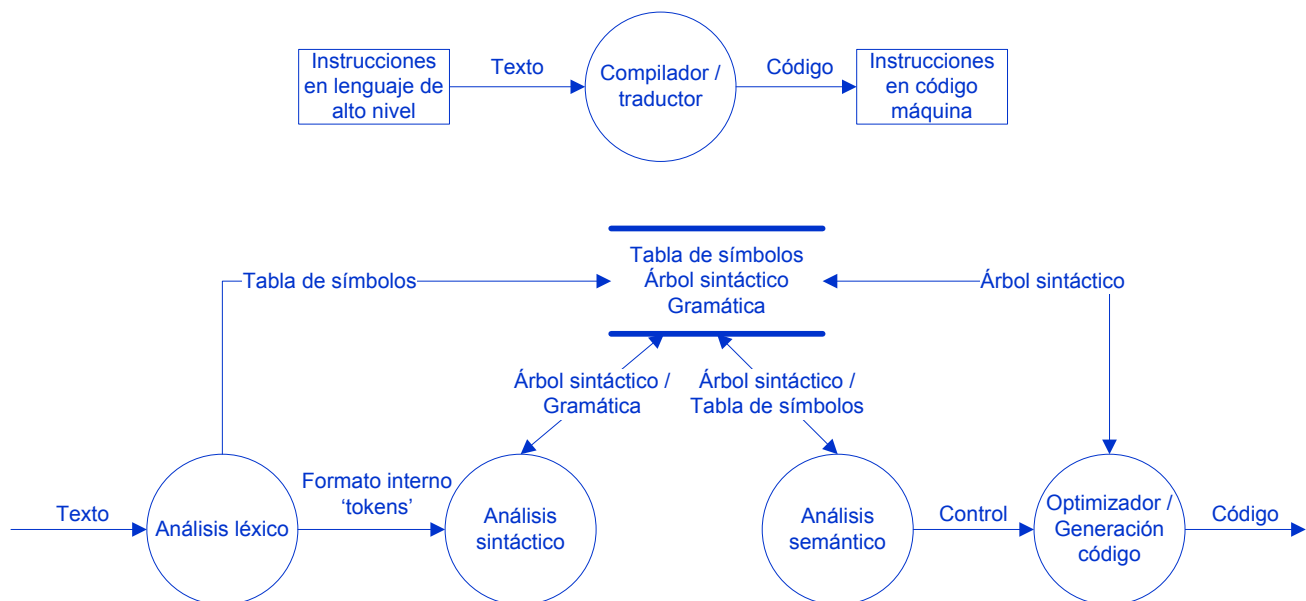
Validar el sistema significa comprobar que el sistema desarrollado cumple con los requisitos previstos desde el principio, especificados por el cliente y recogidos en el documento de análisis. Para ello se aplican estrategias de caja negra a todo el sistema una vez integrado.

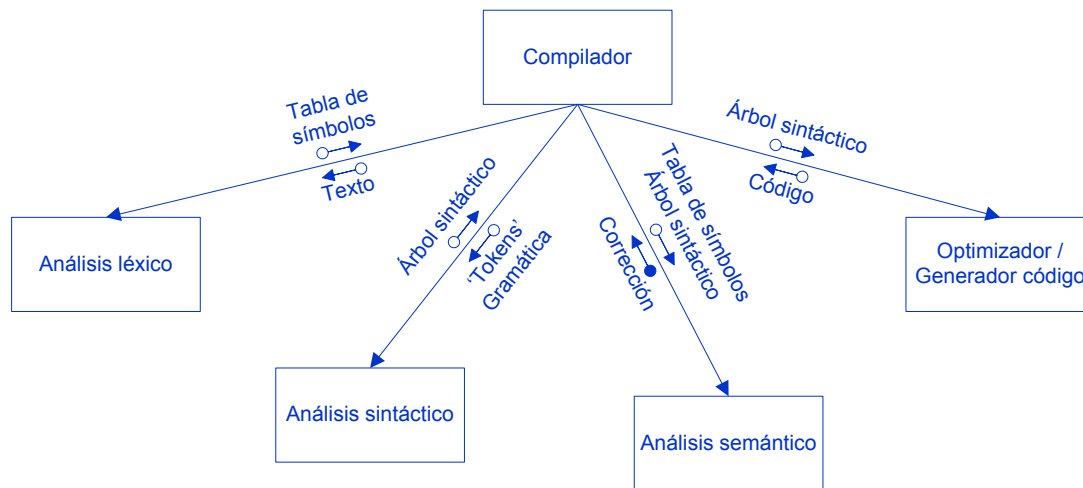
Por último se emplean las pruebas alfa y beta para ver la utilidad que tiene la aplicación a sus usuarios finales.

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

3. Los sistemas de procesamiento de lenguaje aceptan como entrada sentencias en algún lenguaje y generan como salida alguna otra representación del lenguaje de entrada. Un caso particular son los compiladores, que traducen un lenguaje de programación artificial de alto nivel a código máquina (abstracta o de un procesador real). Los principales componentes de un traductor (sea para un compilador u otro procesador de lenguaje) son:
- 1) Un analizador léxico, que toma como entrada los elementos del lenguaje, identifica sus símbolos y los convierte a un formato interno.
 - 2) Una tabla de símbolos, que almacena información sobre los nombres de las entidades (variables, nombres de clases, de objetos, etc.) usadas en el texto que se está traduciendo.
 - 3) Un analizador sintáctico, que comprueba la sintaxis del lenguaje que se está traduciendo. Utiliza una gramática definida y construye un árbol sintáctico.
 - 4) Un árbol sintáctico, que es una estructura interna que representa el texto que se está traduciendo o el programa que se está compilando.
 - 5) Un analizador semántico, que utiliza la información del árbol sintáctico y de la tabla de símbolos para comprobar la corrección semántica del texto en el lenguaje de entrada.
 - 6) Un optimizador, que transforma el árbol sintáctico para mejorar la eficiencia y eliminar redundancias en el código máquina que se genere.
 - 7) Un generador de código, que «recorre» el árbol sintáctico y genera código máquina (traducido).
- a) Represente la arquitectura genérica de un compilador mediante un modelo de flujo de datos (haga el DFD, el correspondiente análisis de flujo de datos y represente la arquitectura con un diagrama de estructura). (2 puntos)

Solución



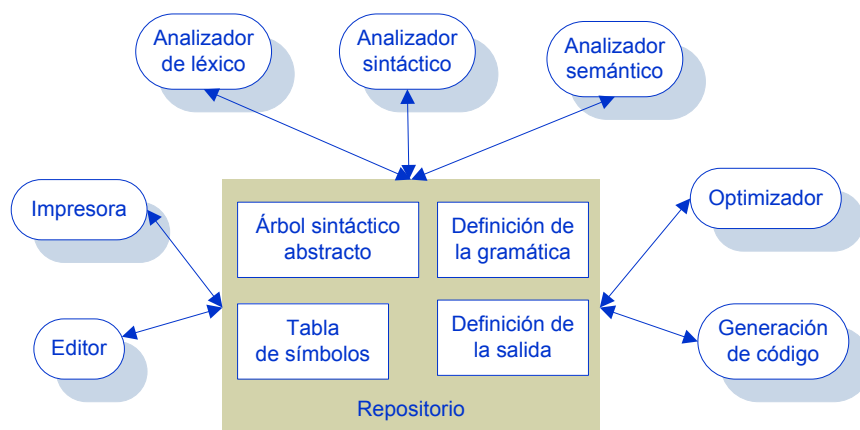


Una decisión fundamental para el diseño de la arquitectura es la estrategia para estructurar el sistema. Se denomina «modelo de repositorio» a la estructura en la que los subsistemas se agrupan en torno a una base de datos compartida, almacén o repositorio. Suponga ahora que el compilador anterior forma parte de un conjunto integrado de herramientas de soporte para la programación. Tales herramientas son el compilador, un sistema de edición estructurado, un depurador interactivo y un programa de impresión. La gramática del lenguaje y el formato de salida del programa son elementos generalmente embebidos en las herramientas pero, por ser comunes, se incluyen en el repositorio junto con los datos comunes del compilador. El repositorio no sólo sirve de almacén, sino también de vía de comunicación entre las herramientas.

- b) **Transforme la arquitectura definida anteriormente para el compilador a un modelo de repositorio al integrarlo con las herramientas mencionadas en un entorno de programación con un editor dirigido por la sintaxis (según se escribe, verifica si la sintaxis es correcta e informa al usuario).** (2 puntos)

Sugerencia: para que se vea más claro y sencillo, no use un diagrama de estructura sino un diagrama de bloques o uno parecido a un DFD.

Solución




c) *¿Qué ventajas tiene la última estructura arquitectónica respecto a, por ejemplo, el editor dirigido por la sintaxis, el depurador interactivo o el sistema de impresión? (1 punto)*

Solución

Como la definición de la gramática está compartida en el repositorio –en lugar de estar embebida en el editor, que supervisa la sintaxis–, se puede comprobar la corrección sintáctica del programa fuente al mismo tiempo que se está escribiendo.

Un depurador interactivo utiliza el programa compilado –lenguaje máquina– y suele permitir hacer un seguimiento en la ejecución, ejecución paso a paso y visualización de los valores de las variables, posiciones de memoria, registros, etc. Esta funcionalidad del depurador es más efectiva cuando el compilador está integrado junto con las otras herramientas –arquitectura en repositorio común–. No lo es tanto si el compilador está aislado de las otras herramientas; en cuyo caso, está más orientado al procesamiento por lotes, en el que los programas son compilados y ejecutados sin la interacción del usuario.

De igual forma, al estar embebido también el formato de salida del programa, el sistema de impresión puede generar, en cualquier momento, listados adecuados que son fáciles de leer.

INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Modelo			
	Junio 2008			
				Ámbito
				CP -- ORIGINAL

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

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)

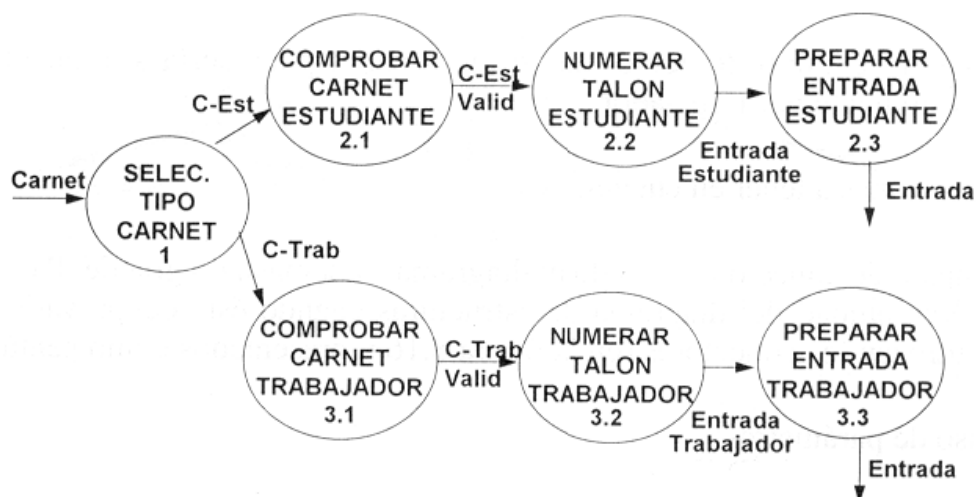
PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. Explique el concepto de abstracción aplicado al diseño de software y sus formas fundamentales de uso.

Solución

Por medio de la abstracción prescindimos de los elementos y utilidades concretas de un sistema software determinado para considerarlos más allá de ese sistema. De esta manera, los elementos que diseñemos serán fácilmente **mantenibles** (se podrán sustituir por otros con mejores prestaciones) y **reutilizables** en otros proyectos similares. Hay tres formas fundamentales de abstracción: abstracciones funcionales, tipos abstractos de datos y máquinas abstractas.

2. La Comunidad Autónoma ha decidido hacerse cargo de la gestión de la piscina «El Remojón». Para ello sólo dejará acceder, de forma gratuita, a dichas instalaciones a aquellos usuarios que sean estudiantes o trabajadores locales. Dependiendo del tipo de usuario (estudiante o trabajador), que se acreditará con un carnet, se realizarán diferentes tratamientos para imprimir la entrada. Suponga que este es el DFD de nivel 2 simplificado para un diseño estructurado:



Se pide:

- Aislar el centro de transformación o el de transacciones. Defina estos conceptos y muestre un diagrama de estructura con cada uno de estos casos. (1 punto)
- Represente el diagrama de estructura correspondiente al diseño estructurado de este ejemplo. (1'5 puntos)

Solución

- Un *centro de transformación* es una operación, acción, proceso o agrupación de ellos, que define la transformación fundamental de un sistema en el que se ha identificado un flujo de información global desde los elementos de entrada hacia los de salida.

Un *centro de transacción* es un proceso que define una bifurcación, a partir de la cual, el flujo de datos global se descompone en líneas separadas. En el enunciado, la selección del carnet es el centro de transacción que activa una u otra línea (transacción) en función del tipo de dato de entrada –el tipo de carnet–.

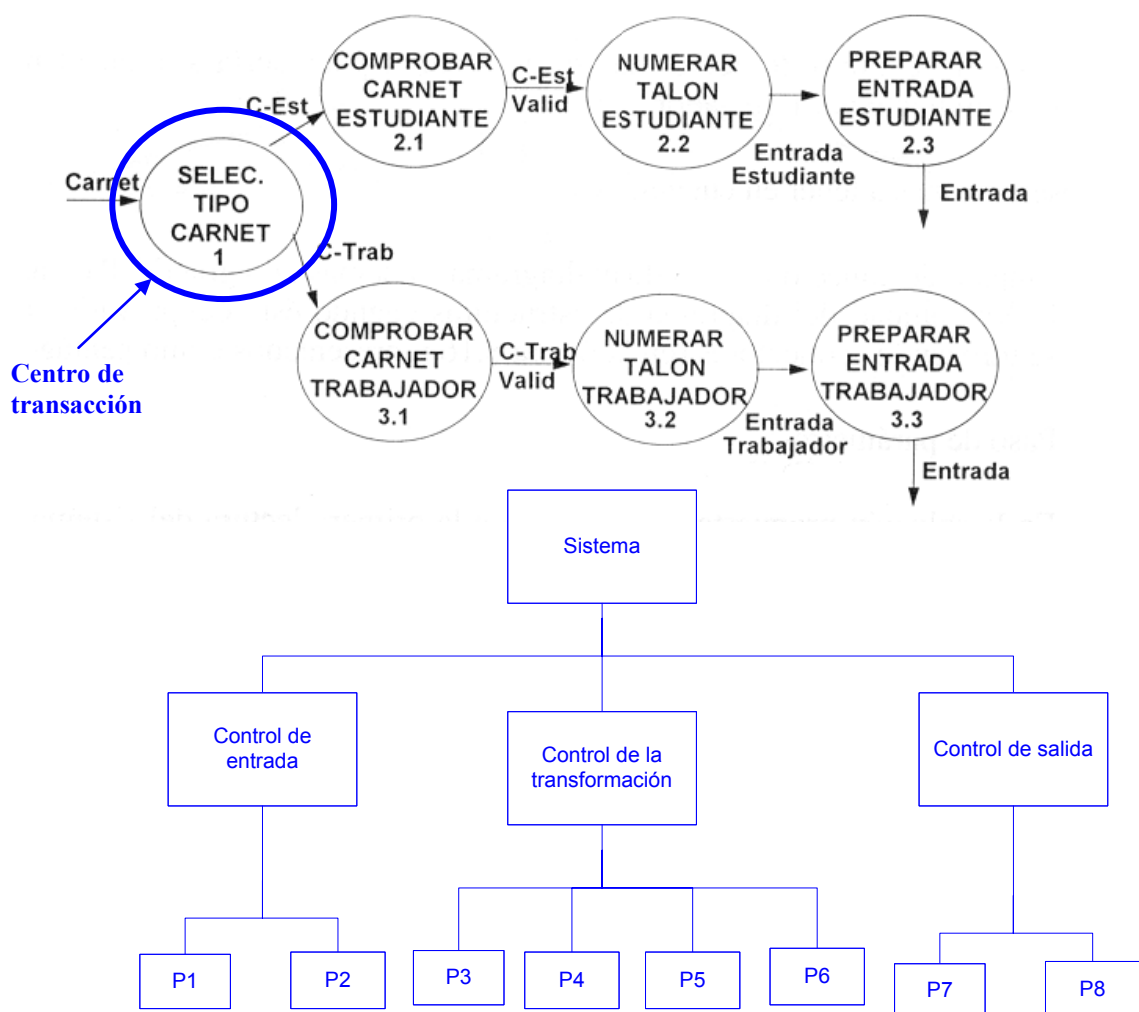


Diagrama de estructura con flujo de transformación.

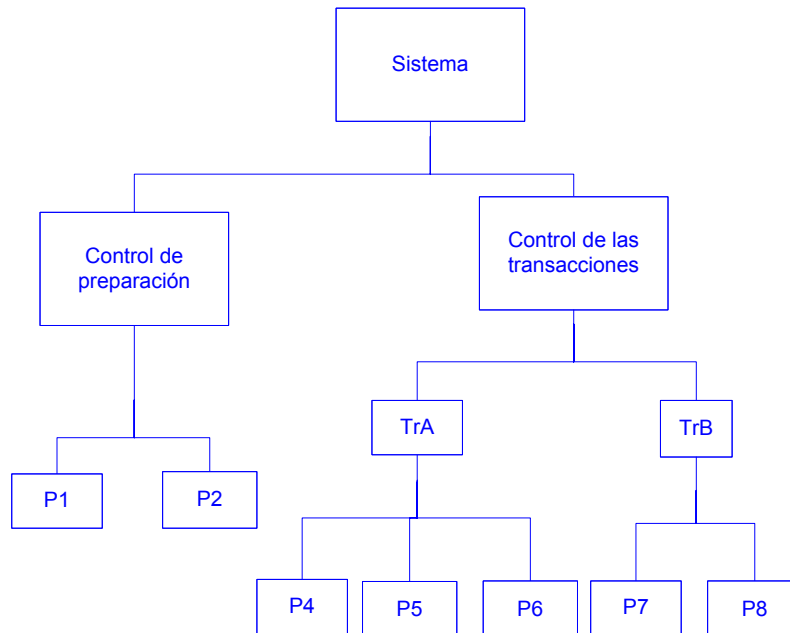


Diagrama de estructura con flujo de transacciones.

b)

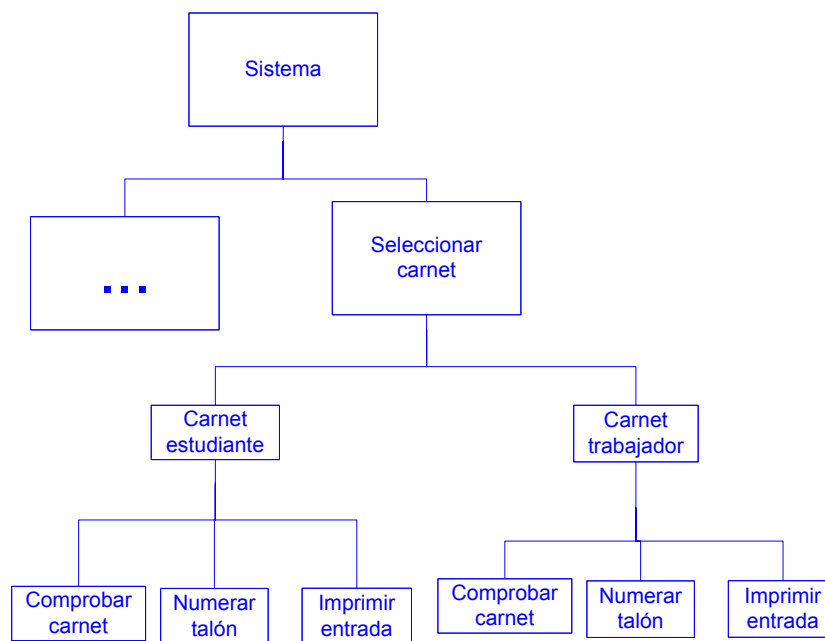


Diagrama de estructura para el control de la entrada a la piscina.

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

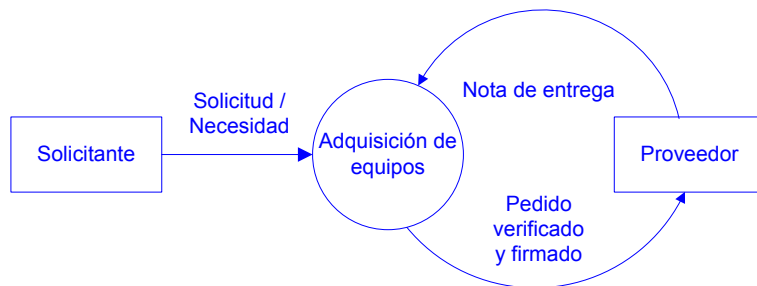
- Se le pide que defina el proceso de adquisición de equipos en una empresa. Esto implica especificar el equipo deseado, buscar y elegir a los proveedores teniendo en cuenta, también, el presupuesto disponible y su aceptación; pedir el equipo, recibirlo a su llegada y, a continuación, verificarlo. La petición del equipo se hará mediante formularios en los que no sólo se reflejarán los detalles del equipo y del proveedor, sino que deberán ir visados para garantizar que hay disponibilidad de fondos y se ha aprobado el gasto. Al emitir el pedido, este se registra para un posterior seguimiento. La recepción de un equipo requiere la verificación de la entrega, instalación, comprobación del funcionamiento especificado, aceptación y registro en la base de datos de inventario.

- a) Resuelva las faltas y ambigüedades que estime. Anótelas e indique las decisiones que ha tomado. (1 punto)
- b) Represente la funcionalidad del proceso mediante diagramas DFD (nivel 0, 1 y 2). (3 puntos)
- c) En el nivel 1, enmarque con una línea discontinua qué parte del proceso decide usted automatizar informáticamente. Explique y argumente su decisión. (1 punto)

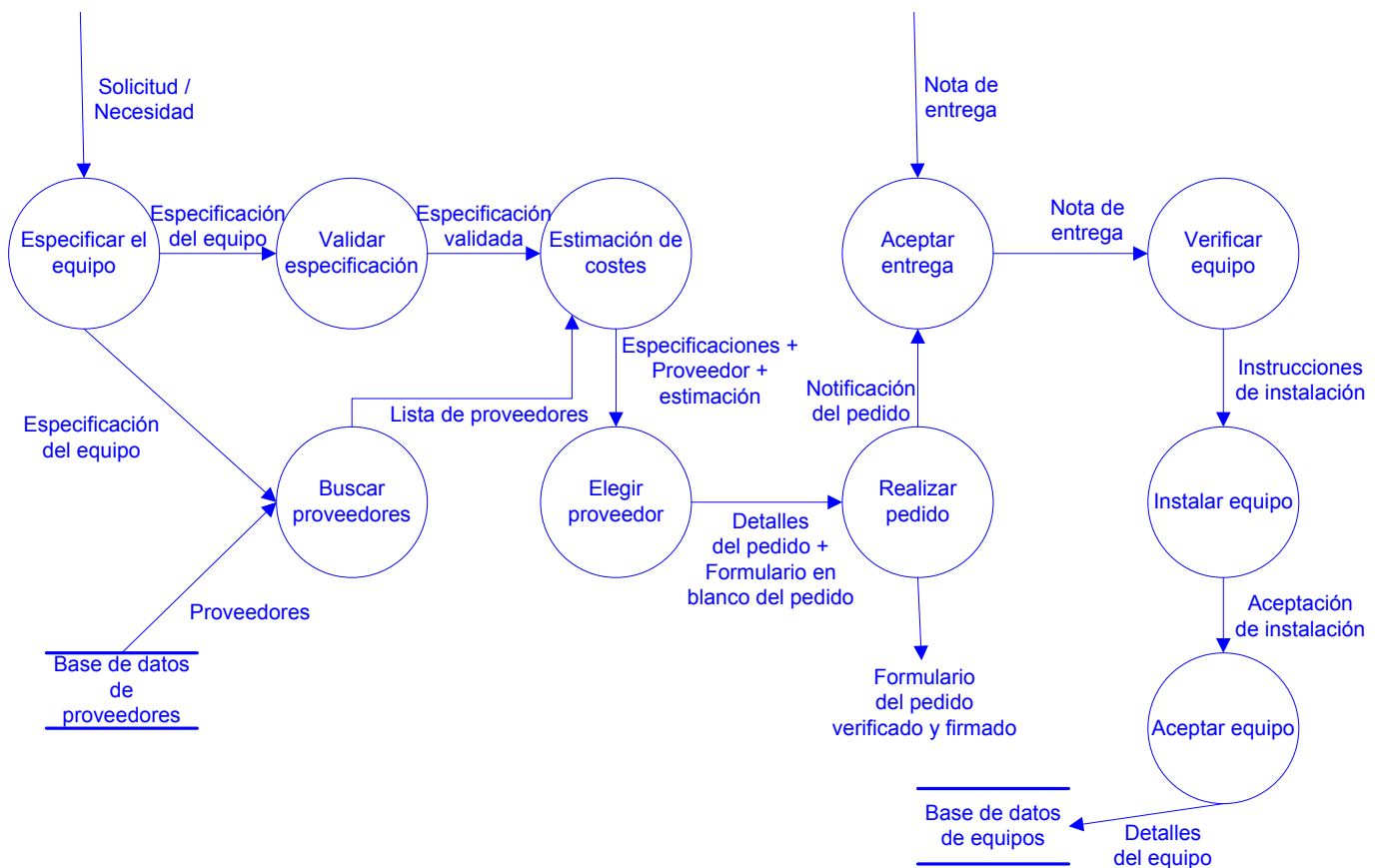
Solución

- a) No queda claro cómo se tiene en cuenta el presupuesto disponible y cuál es el procedimiento de aceptación del pedido y visado de éste. Tampoco queda claro si la evaluación del pedido, su aprobación, registro de éste y el del inventario del equipo –una vez recibido y verificado– se realiza en el propio sistema o bien es una entidad externa como, por ejemplo, ‘Gerencia. Gestión económica e inventario’ la que lo hace.

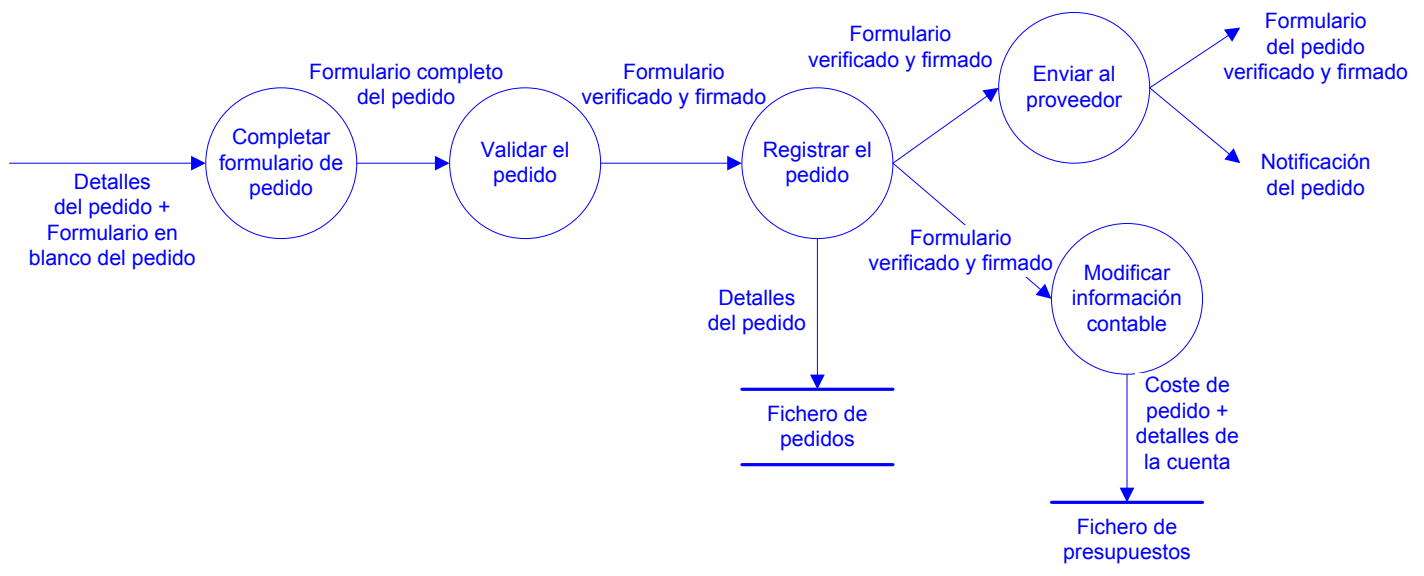
b)



DFD de contexto

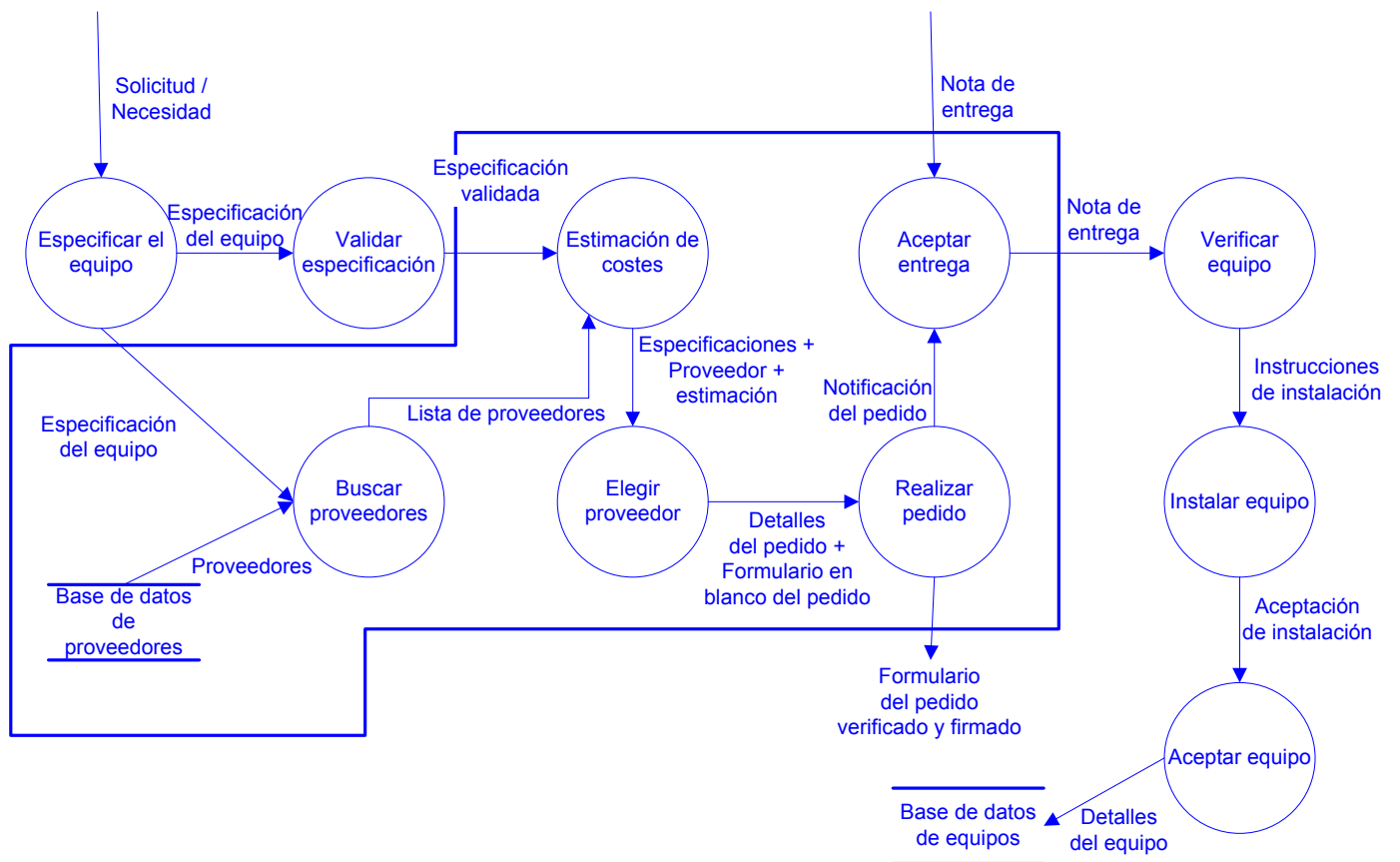



DFD de nivel 1



DFD nivel 2. Explosión de 'Realizar pedido'

c) Se sugiere la parte que se automatiza con un recuadro de línea continua más gruesa.



INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Modelo			
	Septiembre 2008			
				Ámbito
				NACIONAL – UE ORIGINAL

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

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.

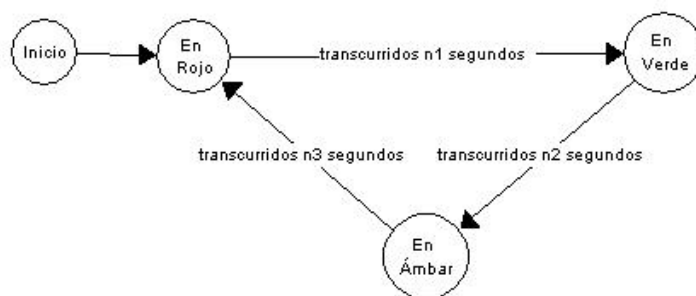
Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm)

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

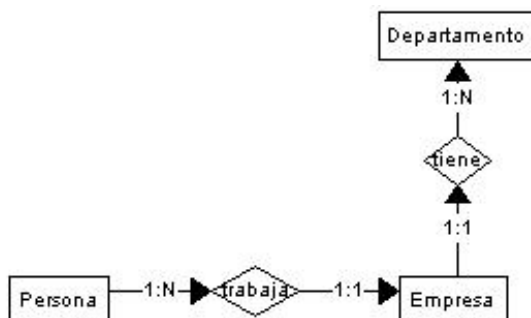
1. Los Diagramas de Transición de Estados (DTE) y los Diagramas Entidad Relación (DER) son notaciones para describir distintos aspectos de un sistema software. ¿Cuáles son estos aspectos? Razone la contestación proponiendo un ejemplo sencillo de aplicación de cada notación.

Solución

Los DTEs sirven para modelar el comportamiento temporal de un sistema software. Por ejemplo, el siguiente diagrama representa la secuencia de colores de un semáforo.



Los DERs permiten modelar la estructura de los datos que maneja una aplicación informática. Por ejemplo, el siguiente diagrama muestra el tipo de elementos que manipula un sistema de nóminas.



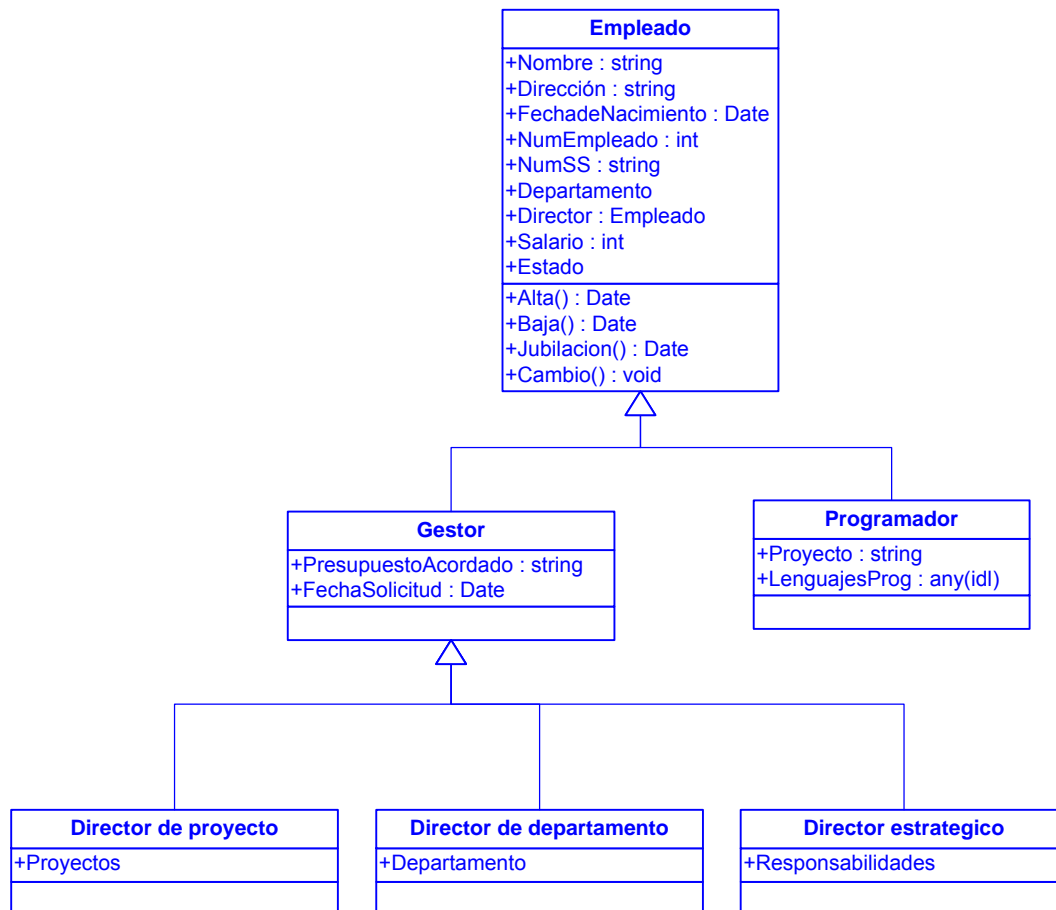
OBSERVACIÓN: un ejercicio interesante para entender los puntos fuertes y las limitaciones de cada notación sería tratar de modelar cada ejemplo intercambiando la notación propuesta en esta solución.

2. A partir de la siguiente descripción:

«Los empleados de una organización de desarrollo de software pueden ser gestores o programadores. Los programadores se caracterizan por la lista de lenguajes de programación para los que están capacitados para hacer desarrollos y por el proyecto en el que están trabajando. Los gestores, por el contrario, se diferencian por el presupuesto de los trabajos que gestionan y por la fecha de solicitud de dichos trabajos. Los gestores pueden ser, a su vez, directores de proyecto –caracterizados por el proyecto que gestionan–, directores de departamento o directores estratégicos –caracterizados por determinadas responsabilidades en las áreas de negocio de la organización o de la unidad a la que pertenecen–».

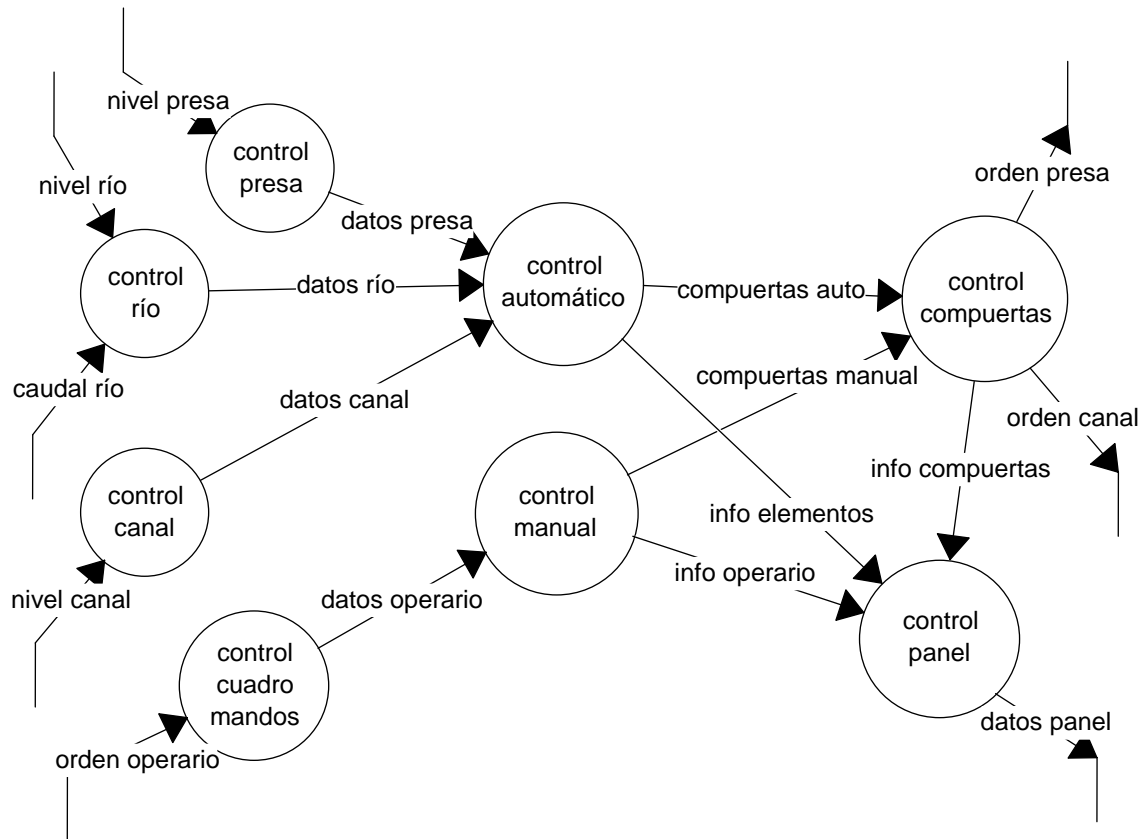
Construya un modelo de objetos solamente de los empleados.

Solución



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

3. La figura que se muestra a continuación corresponde al diagrama de flujo de datos de un **sistema de control de desbordamiento de una presa**. Realiza un control automático coordinado de 2 compuertas: la de la presa y la de un canal de riego abastecido por el río al que vierte agua el embalse. Además, muestra en un panel de control un diagrama que representa todos los elementos del sistema y su estado instantáneo. El control de las compuertas se basa en datos proporcionados por sensores de nivel y de caudal situados en el embalse, el río y el canal de riego. También se puede ejercer un control manual de las compuertas mediante un cuadro de mandos de operario.



Los objetivos que debe cumplir el sistema son los siguientes, **de mayor a menor prioridad**:

- atender las peticiones del operario
- evitar desbordamientos de la presa
- evitar desbordamientos del río
- evitar desbordamientos del canal de riego
- garantizar agua en el canal de riego

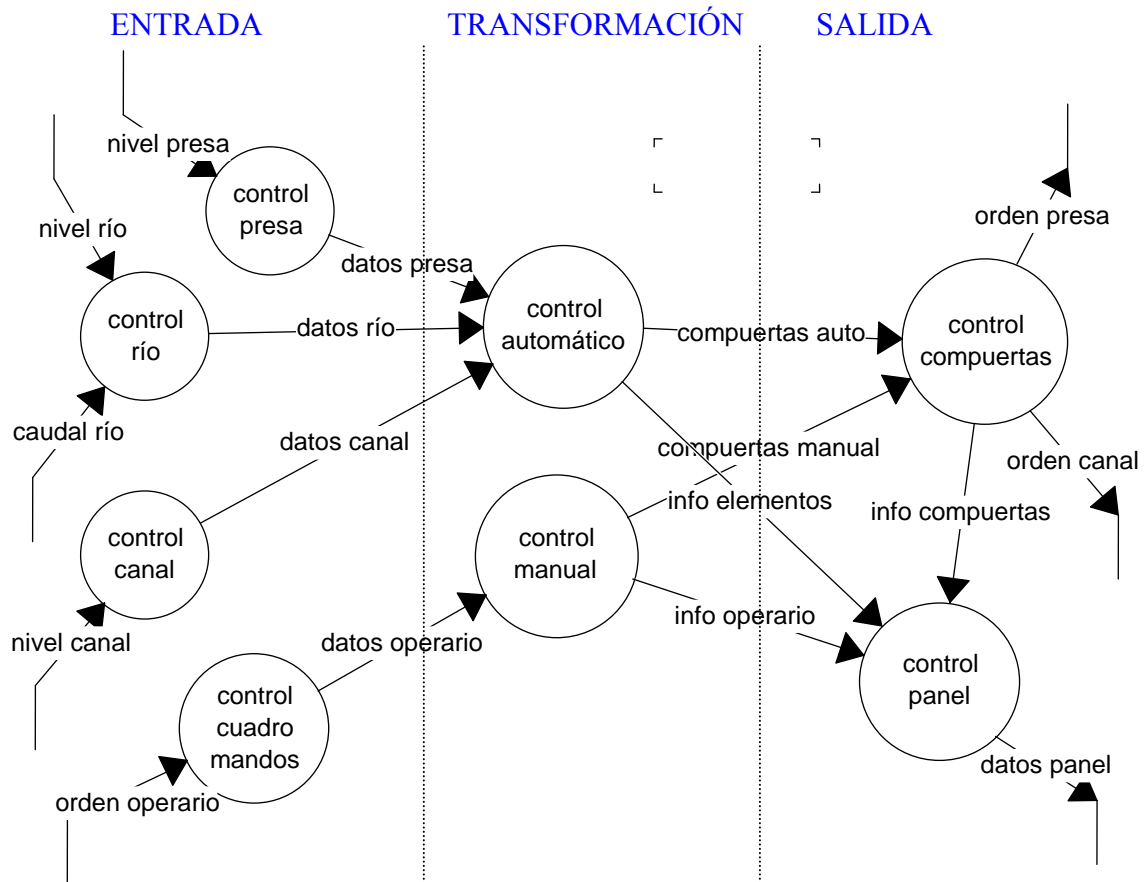
Se pide:

- Utilizando la técnica de diseño estructurado, realice el diagrama de estructura del sistema a partir del anterior DFD.
- Proponga un algoritmo, utilizando pseudocódigo, para el diseño procedimental del módulo “control de transformación” resultante del paso 1. Haga lo mismo con los módulos que cuelgan de él. Tenga para ello en cuenta los objetivos del sistema y su prioridad.

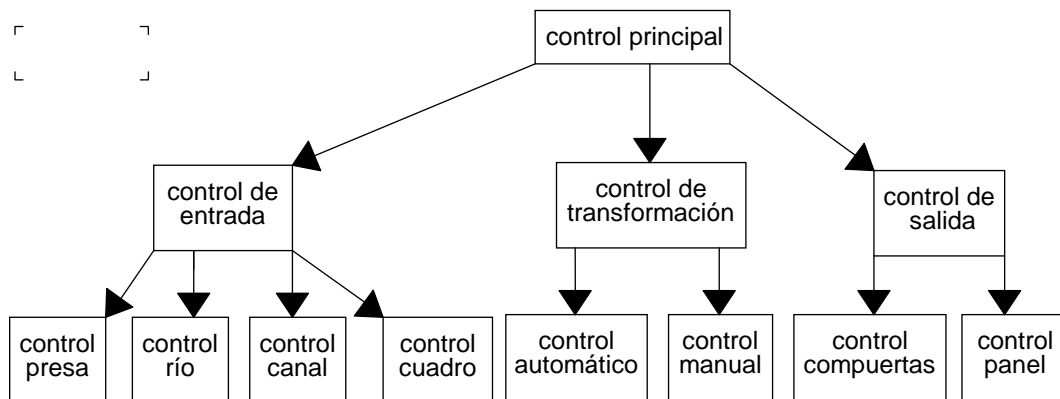
Solución

- El diagrama de estructura añade al DFD una jerarquía de control. La técnica de diseño estructurado recomienda realizar análisis de flujo de transformación o de flujo de transacción para establecer

dicha jerarquía. En nuestro caso el análisis de flujo apropiado es el de transformación, siendo fácilmente identificables los flujos de información de entrada, transformación y salida del sistema:



El diagrama de estructura será:



b) “Control de transformación”:

```


REPETIR
    MIENTRAS no lleguen "datos operario"
        Realizar "control automático"
    FIN-MIENTRAS
    Realizar "control manual"
HASTA apagado del sistema
  
```

“control automático”:

```
SI "datos presa" = alerta desbordamiento presa ENTONCES
    abrir compuerta presa y abrir compuerta canal
    SI NO, SI "datos rio"= alerta desbordamiento río ENTONCES
        cerrar compuerta presa y abrir compuerta canal
        SI NO, SI "datos canal"=alerta desbordamiento canal ENTONCES
            cerrar compuerta canal
            SI NO abrir compuerta canal
        FIN-SI
    FIN-SI
FIN-SI
```

“control manual”:

```
CASO "dato operario"
    SI-ES abrir compuerta presa: ENTONCES abrir compuerta presa
    SI-ES abrir compuerta canal: ENTONCES abrir compuerta canal
    SI-ES cerrar compuerta presa: ENTONCES cerrar compuerta presa
    SI-ES cerrar compuerta canal: ENTONCES cerrar compuerta canal
FIN-CASO
```

INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Modelo			
	Septiembre 2008			
				Ámbito
				NACIONAL – UE RESERVA

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

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)

- Según Ian Sommerville, el objetivo final de un desarrollo basado en prototipo rápido es *«trabajar con el cliente en la construcción de un producto final a partir de una especificación simplificada. Se debería partir de una buena comprensión del sistema inicial y sus requisitos e ir añadiendo nuevas prestaciones según sean propuestos por el cliente.»* Desde el punto de vista de la gestión y de la ingeniería, se producen estos inconvenientes:
 - Si una ventaja es la pronta entrega de versiones al cliente, aunque con funcionalidad limitada, esto puede producir que no resulte rentable generar la documentación necesaria para un conocimiento adecuado de las versiones intermedias; con la consiguiente falta de visibilidad del progreso real del desarrollo.
 - El añadir nuevos requisitos y funcionalidad en las sucesivas versiones, facilita que la estructura de la aplicación se degrade o se corrompa. El resultado final puede tener una estructura deficiente; es decir, el funcionamiento definido para la descomposición modular inicial puede no ser coherente con el funcionamiento de los componentes finales –sobre todo si no se refactoriza (reingeniería)–, aunque las versiones y el resultado final se hayan validado correctamente.
 - En el desarrollo completo de sistemas grandes –o de ciclo de vida muy largo–, en el que lo habitual es que el trabajo se distribuya entre equipos independientes, el enfoque evolutivo obliga a mantener una vinculación y comunicación muy estrecha entre los desarrolladores y con el cliente; lo cual es difícil realizar. Por consiguiente, el modelo evolutivo no es aconsejable para este tipo de sistemas.

A la vista de lo anterior, argumente cómo se ve afectado el mantenimiento del producto final.

Solución

En el caso de la falta de documentación de las versiones, la repercusión para el mantenimiento es pequeña; a no ser que dicha documentación pueda ser de ayuda para una refactorización de la versión final y la generación de la documentación definitiva.

En cuanto a la degradación de la estructura, parece evidente que si la adición de funcionalidad se hace «*parcheando*» el diseño inicial y, por motivos de plazos o presupuesto, no se rediseña el resultado final, éste puede validarse correctamente, pero el mantenimiento puede ser difícil por defectos en el diseño – dependencia modular, poco comprensible o adaptabilidad escasa–.

Por último, en el caso de sistemas de gran tamaño –proclives a tener un funcionamiento complejo–, su mantenimiento suele ser más difícil que el de sistemas más simples. Además, las exigencias de este tipo de sistemas respecto a la estabilidad de los requisitos y la coordinación entre equipos de desarrollo –ambas contrarias a la naturaleza del enfoque evolutivo– hacen que, si no se cumplen dichas exigencias, el mantenimiento sea aún más difícil.

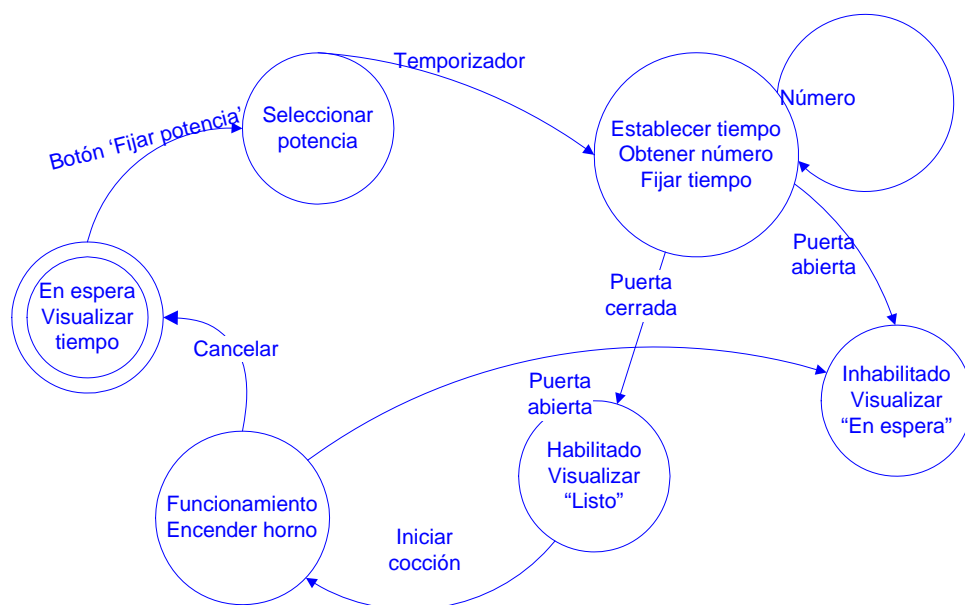
2. Suponga que el modelo simplificado de uso de un horno microondas implica estas acciones:

1. Seleccionar el nivel de potencia (media o máxima).
2. Introducir el tiempo de cocción.
3. Pulsar el botón de inicio y la comida se cocina durante el tiempo establecido.

El aparato tiene botones para fijar la potencia, el temporizador e iniciar el sistema. Por seguridad, el horno no debería funcionar con la puerta abierta. Cuando se completa la cocción, suena un timbre. El aparato tiene una pantalla alfanumérica para representar información de selección, progreso, alerta y precaución.

Construya un Diagrama de Transición de Estados para representar este funcionamiento.

Solución



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

3. **Modele con un diagrama de objetos lo que se indica en el siguiente enunciado:**

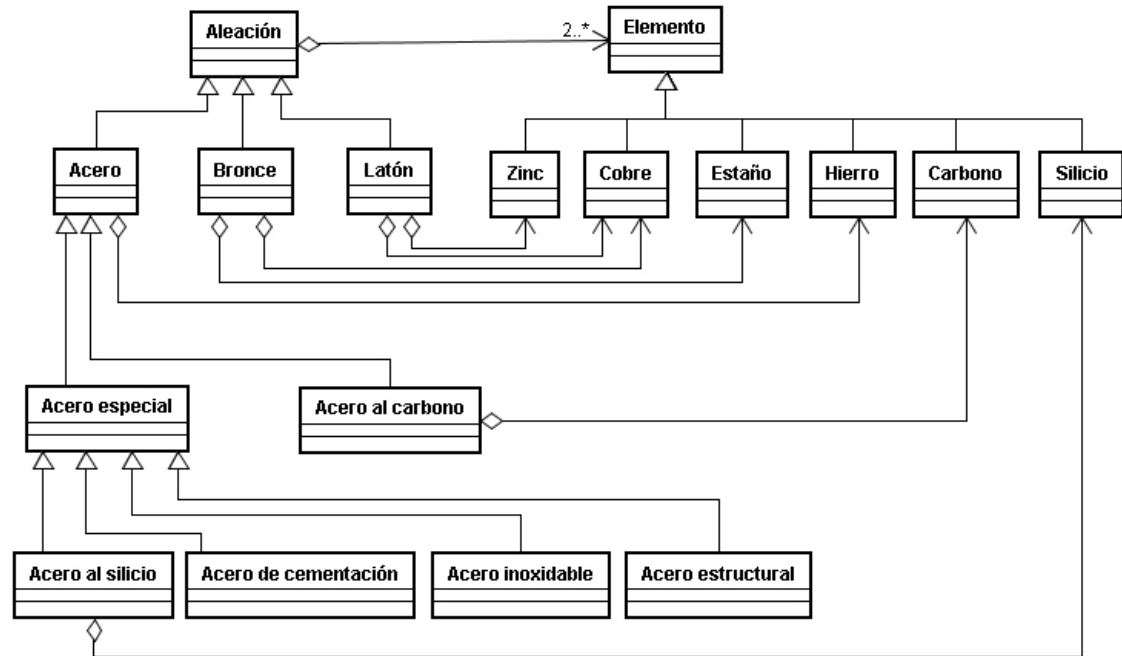
Por aleación se entiende la unión homogénea de dos o más elementos. Generalmente, se dice que el acero es una aleación de hierro y carbono. Sin embargo, esta definición se restringe a los “aceros al carbono”, ya que existen otros tipos de acero con composiciones muy diversas que reciben denominaciones específicas en virtud:


- de los elementos que, además del hierro, predominan en su composición (por ejemplo, aceros al silicio).
- de su susceptibilidad a ciertos tratamientos (por ejemplo, aceros de cementación)

- de alguna característica potenciada (por ejemplo, aceros inoxidables)
- en función de su uso (por ejemplo, aceros estructurales).

Usualmente, estas otras aleaciones de hierro se engloban bajo la denominación genérica de “aceros especiales”. Además del acero, otras aleaciones muy comunes son el bronce, formado por cobre y estaño, y el latón, compuesto por cobre y zinc.

Solución



INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Modelo			
	Septiembre 2008			
		Ámbito		
		CENTROS PENITENCIARIOS ORIGINAL		

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

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 metodología llamada 'programación estructurada', ¿qué técnica utiliza en la fase de diseño?

Solución

La metodología de programación conocida como 'programación estructurada' se empezó a utilizar a primeros de los años 70, siendo los trabajos clásicos de Dijkstra, Dahl y Wirth los que le dieron popularidad. En la fase de diseño de esta metodología se utiliza una técnica (muy intuitiva) llamada 'desarrollo por refinamiento progresivo' y consiste en descomponer el sistema desde un punto de vista funcional y ver el sistema entero como una operación (función) global y, a partir de ahí, ir descomponiéndolo siempre en funciones más sencillas.

Véase sección 4.2.1 del libro de texto.

2. Razone qué ventajas puede tener la técnica de integración llamada 'sándwich'.

Solución

La técnica 'mixta' o 'sándwich', al integrar tanto descendente -con los módulos de nivel alto- y ascendente -con los de nivel bajo-, nos ofrece las ventajas de ambas técnicas al mismo tiempo. Permite ver desde un principio las posibilidades de la aplicación al ir integrando de forma descendente y, al mismo tiempo, se puede trabajar en paralelo con otros desarrolladores que estarán integrando de forma ascendente; empezando con los módulos de nivel bajo.

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

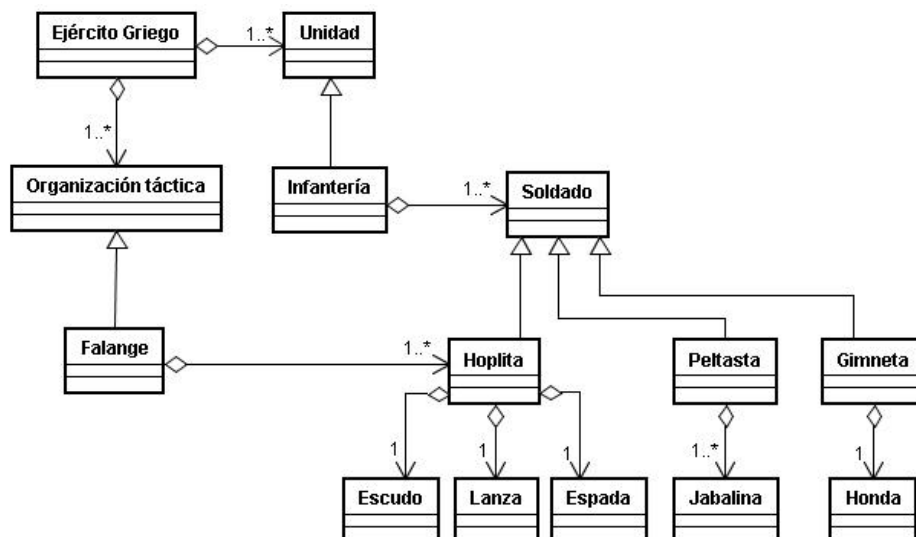
3. *Modele, con un diagrama de objetos, la composición de un ejército griego según el siguiente enunciado:*


En la antigua Grecia, la principal unidad del ejército era la infantería, que estaba compuesta por tres tipos de soldados: 1) los hoplitas, pesadamente equipados con un escudo, una lanza, y una espada; 2) los

peltastas, que portaban un equipamiento más ligero compuesto por varias jabalinas; y 3) los gimnetas, que llevaban una honda.

La principal organización táctica para la guerra era la falange, que luego fue imitada por varias civilizaciones mediterráneas. La falange estaba formada por una profunda fila de hoplitas.

Solución



INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Modelo			
	Junio 2009			
				Ámbito
				NACIONAL 1ª SEMANA

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

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)

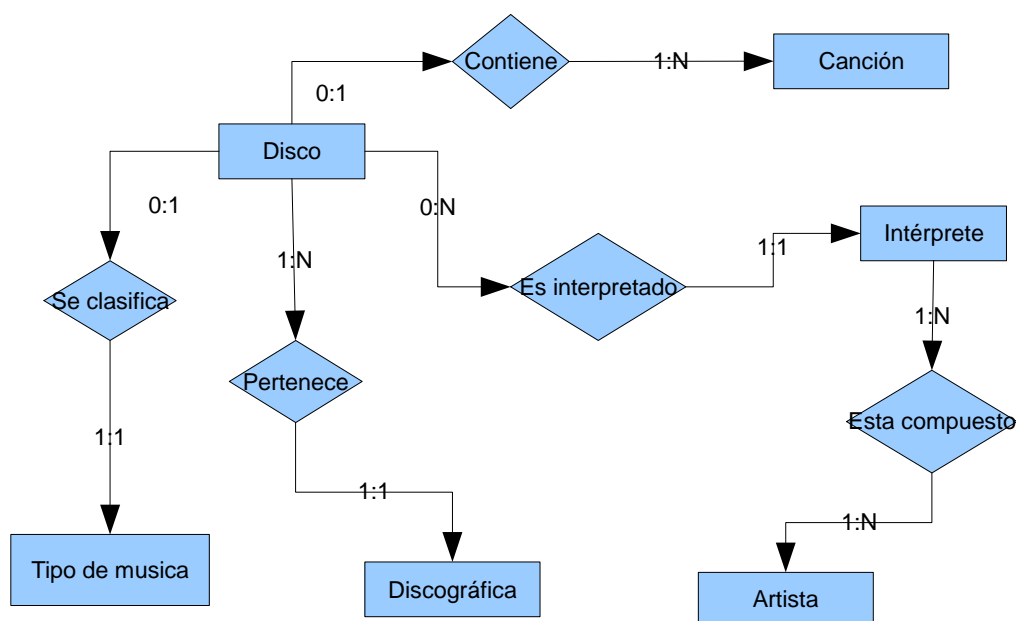
PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

- La gestión del catálogo de discos de una biblioteca se debe basar en la siguiente especificación:

Un disco tiene un título y un intérprete. El intérprete puede ser un artista sólo o un grupo (conjunto de artistas). El disco pertenece a una discográfica y contiene una serie de canciones. Así mismo, cada disco puede ser clasificado según un tipo de música.

Modele el enunciado anterior mediante un **Diagrama Entidad-Relación**. Describa los principales datos utilizando la notación del **Diccionario de Datos**.

Solución



Diccionario de datos:

DISCO = Nombre + INTERPRETE + TIPO-DE-MÚSICA + DISCOGRÁFICA + LISTA-DE-CANCIONES + año

INTERPRETE = ARTISTA

ARTISTA = {nombre}

DISCOGRÁFICA = nombre


TIPO-DE-MÚSICA = [jazz | clásica | pop | rock | heavy | dance]

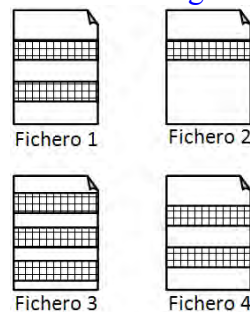
CANCIÓN = nombre + ARTISTA



LISTA-DE-CANCIONES = {CANCIÓN}

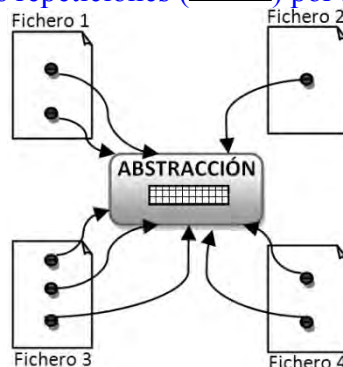
2. Una práctica muy extendida para reutilizar software es “cortar y pegar” fragmentos de código. Esto puede llevar a que un trozo de código se encuentre repetido muchas veces en un programa. ¿Cómo afecta esto al mantenimiento del programa? ¿Cómo podrían solventarse los efectos negativos del código duplicado?

Solución

La duplicidad de código perjudica la adaptabilidad de un programa. Por ejemplo, la siguiente figura representa un programa compuesto por 4 ficheros que contienen cierto código repetido (representado por ). Si durante el mantenimiento del programa se decide modificar dicho código repetido, habrá que realizar el cambio en todos los lugares donde se encuentra el código (en el ejemplo, habría que repetir el cambio en 8 puntos). Este trabajo no sólo es tedioso, si no que puede llevar a inconsistencias cuando los cambios no se aplican a todos los lugares donde el código está repetido.



Como indica la siguiente figura, estos problemas podrían evitarse encapsulando el código repetido en algún tipo de abstracción y sustituyendo las repeticiones () por invocaciones () a la nueva abstracción.

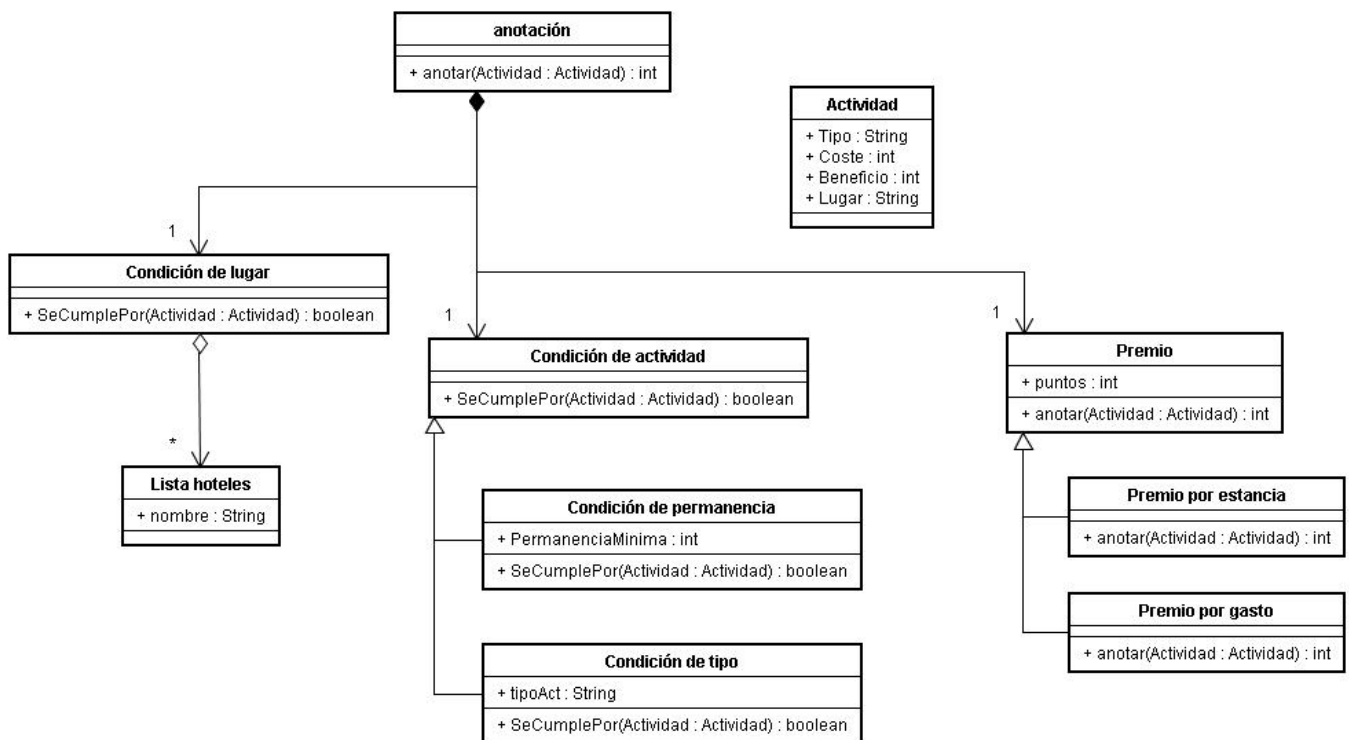



SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

3. Una cadena hotelera ofrece casi todos sus servicios a través de un portal Web. Un mecanismo de fidelización utilizado con frecuencia consiste en que el cliente va acumulando puntos en la medida en que consume los servicios de la empresa, dependiendo de qué actividades realice y en qué condiciones lo haga. Periódicamente, la cadena confecciona ofertas que sitúa en su Web; las cuales son canjeables por los puntos que un cliente tenga acumulados. El módulo de “Fidelización y Gestión de Puntos del Cliente”, básicamente, lo que hace es contabilizar los puntos generados por las ‘actividades’ –consumos de determinados servicios— y anotarlos en la cuenta del cliente. Una Actividad se caracteriza por el ‘tipo’, ‘coste’ del servicio, ‘beneficio’ y ‘Lugar’ donde se consume. Una ‘anotación’ se compone de tres elementos: una ‘condición de lugar’, una ‘condición de actividad’ y el ‘premio’ por consumo. La condición de lugar comprueba si la actividad se realiza en el establecimiento o Lugar adecuado para anotar puntos. Dicha comprobación consulta una lista de establecimientos adheridos a la cadena hotelera. La condición de actividad es o bien la comprobación de que la permanencia en el hotel es mayor que un determinado valor, o bien que el tipo de la actividad es el adecuado para puntuar. Una vez que se cumplen las dos condiciones anteriores, se hace el cálculo del premio. El premio se calcula de dos formas excluyentes: o es un ‘premio por día de estancia’ o es un ‘premio por cantidad gastada’.

Represente la descripción del módulo anterior mediante un Diagrama de Objetos.

Solución



INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Modelo			
	Junio 2009			
		Ámbito		
		NACIONAL Y U.E. 2ª SEMANA		

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

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. Describa los documentos o productos resultantes de cada una de las fases del modelo de ciclo de vida en cascada.

Solución

Página 12 de la bibliografía básica.

- Documento de Requisitos de Software (SRD): tras la fase de análisis. Es una especificación precisa y completa de lo que debe hacer el sistema.
 - Documento de Diseño del Software (SDD): tras la fase de diseño. Estructura del sistema, con su descomposición en subsistemas y las relaciones entre ellos.
 - Código Fuente: tras la codificación. Programas fuente convenientemente comentados.
 - Sistema Software: tras la integración. Es el ejecutable junto con las pruebas de validación.
 - Documentos de cambios o mantenimiento: el mantenimiento se debe realizar de manera organizada, contabilizando tanto los informes del problema como los del cambio realizado.
2. Varios módulos de una aplicación utilizan la información de un fichero que está en un disco. Para usarlo, los módulos invocan los servicios que provee la controladora del dispositivo –los cuales trabajan directamente con las características físicas del disco—. Por ejemplo, para escribir en el fichero, los módulos invocan la instrucción imaginaria del microcontrolador del disco: WRITE_STRING(DiskID, Cyl, Sector, Pos, NBytes, "String"). ¿Qué tipo de acoplamiento presenta este caso? ¿Qué consecuencias tiene este uso respecto a la reusabilidad y a la mantenibilidad? ¿Cómo cambiaría esta situación?

Solución

Página 152 de bibliografía básica.

Éste es un claro ejemplo de acoplamiento externo, en el que varios módulos comparten información que reside en un dispositivo externo a ellos y a la que acceden directamente a través del ‘hardware’ que controla el dispositivo.

Esta práctica obliga a realizar todas las operaciones a través de las rutinas de servicio del propio dispositivo. Al ser, dichas rutinas, específicas del dispositivo; la posibilidad de reutilización del código –reusabilidad— en cualquier otro dispositivo que no sea idéntico, está –obviamente— disminuido o anulado. En cuanto a la mantenibilidad –o facilidad y posibilidad de hacer modificaciones en el código—, también se ve gravemente afectada; puesto que, si se cambia algo del dispositivo externo, obliga a cambiar todas las partes del código en las que se utilice dicho dispositivo.

La manera de evitar todo esto es mediante la abstracción y el encapsulamiento de todas estas operaciones en un módulo o paquete que se constituya como un controlador abstracto y genérico del dispositivo –disco— y que contenga, bajo invocaciones genéricas a los antiguos servicios específicos del dispositivo, las rutinas concretas de operación del dispositivo. Así, si se produce algún cambio, sólo será necesario hacer las modificaciones en el ‘controlador abstracto’.

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

3. *Un supermercado desea implantar un sistema de cobro automático, de forma que sean los propios clientes quienes pasen los productos por el lector de código de barras y paguen introduciendo su tarjeta de crédito en una ranura, tras lo que recibirán el comprobante de la compra. El cliente podrá cancelar el proceso en cualquier momento, pero no una vez aceptado el pago.*

Proponga, mediante lenguaje natural un sistema sencillo que resuelva el sistema deseado. Realice el diagrama de transición de estados del sistema propuesto.

Solución

En cada puesto de auto-pago habrá un terminal de punto de venta (TPV) con los siguientes elementos:

- Pantalla táctil
- Lector de código de barras
- Lector de tarjetas de banda magnética
- Impresora de recibos

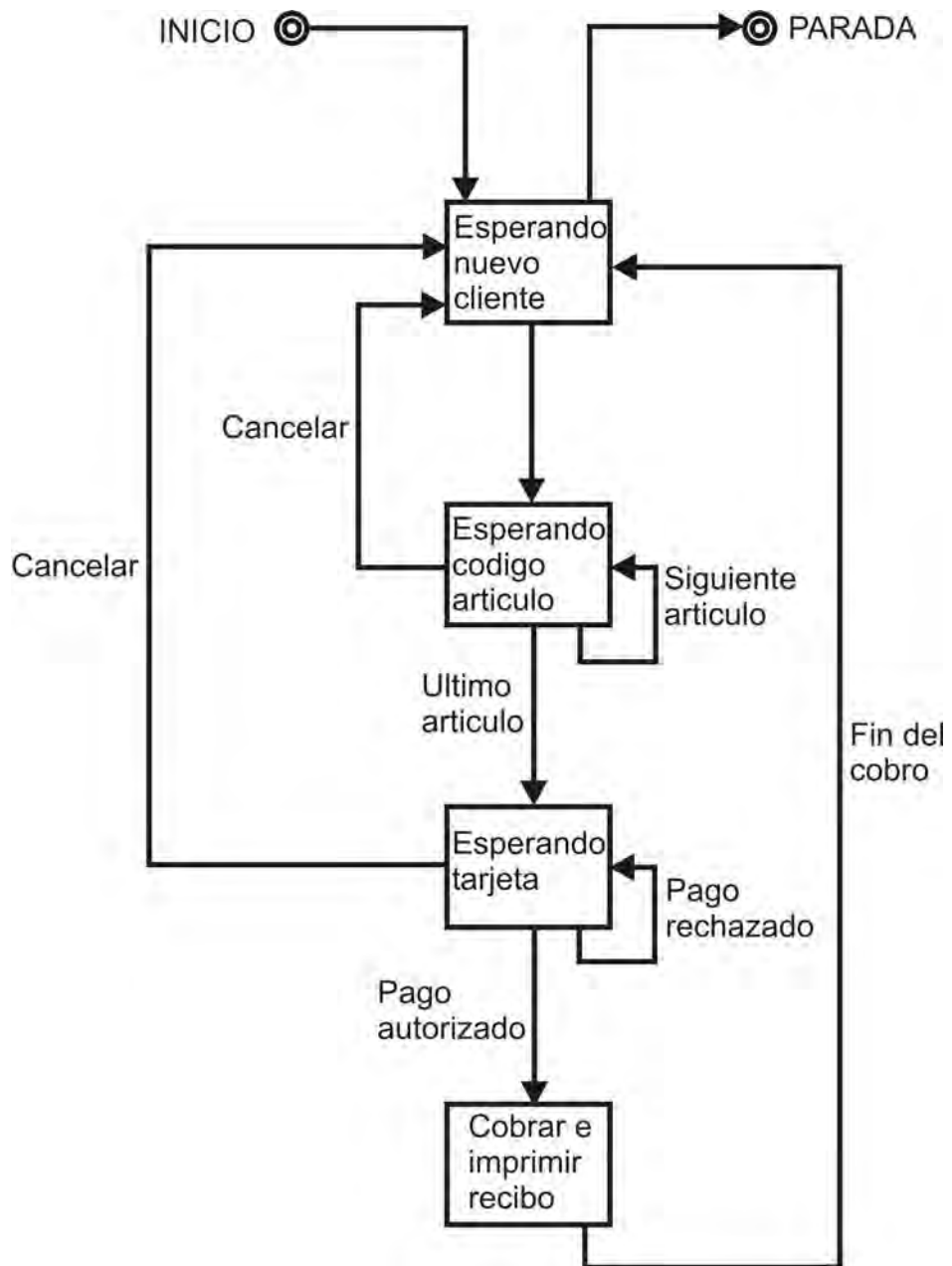
Cada TPV estará conectado al sistema que gestiona la base de datos de existencias, referencias y precios de los productos, y gestionará la autorización para el pago con tarjeta de crédito.


Cuando llegue un nuevo cliente deberá pulsar en “cliente nuevo” en la pantalla. A continuación se le indicará que comience a pasar los productos por el lector de códigos de barras. En la pantalla se presentará la lista de productos leídos con su referencia y precio. En cualquier momento el cliente podrá pulsar “cancelar” o “último artículo”.

“Cancelar” anula todo el proceso de pago; no se puede cancelar el último artículo que se ha detectado sino que se deberá empezar de nuevo a pasar todos los productos por el lector.

Tras pulsar “último artículo” la pantalla mostrará el mensaje “introduzca tarjeta de crédito”. Si el pago es autorizado se imprimirá el recibo y el proceso habrá finalizado. Si no se autoriza el pago con la tarjeta introducida, se mostrará el mensaje “tarjeta no válida”, pudiendo el cliente introducir otra tarjeta o cancelar el proceso.

A continuación se muestra el Diagrama de Transición de Estados del sistema descrito:



INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Modelo			
	Septiembre 2009			
				Ámbito
				NACIONAL - UE ORIGINAL

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

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)

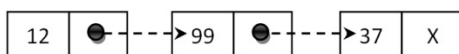
PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. Describa la notación DFD (Diagrama de Flujo de Datos). ¿Qué aspectos de la aplicación modela esta notación?

Solución

Página 52 del texto básico para la asignatura; epígrafe 2.3.2.

2. La lista enlazada es una de las estructuras de datos más utilizadas en programación. Básicamente, se trata de un conjunto de nodos enlazados. Cada nodo N dispone de dos campos; uno que contiene un elemento de cierto tipo (en este enunciado supondremos que contiene números enteros) y otro campo que puede: (a) contener otro nodo N' ó (b) si el nodo N es el último elemento de la lista, no contener nada. Por ejemplo, la siguiente lista [12, 99, 37] se implementaría con una lista enlazada como:



Describa la estructura lista enlazada mediante **a)** Un Diagrama de Objetos y **b)** La notación de Diccionario de Datos.

Solución



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

3. *Su empresa va a acometer el desarrollo de un servicio de Internet para el área del negocio hotelero. En concreto, un sistema basado en Web de reservas on-line de alojamientos turísticos. El sistema tendrá dos ‘vistas’:*
- *La “Operativa de Reservas” dará servicios a “Usuarios” –visitantes de la Web interesados por el alojamiento y la disponibilidad—, a “Agencias/Clientes” –operadores que tienen un cupo asignado durante un período de tiempo y negocian con él intermediando con el “usuario”— y, además, con ella se realizan los “Cobros” mediante TPV virtual –terminal de punto de venta— bajo servidor seguro.*
 - *La otra vista es la de “Gestión del Sistema” que permitirá al establecimiento hotelero o administrador de la Web gestionar la disponibilidad, las tarifas, los cupos, las agencias/clientes, las reservas, hacer seguimiento, consultas, estadísticas, etc.*

Las características principales del sistema son:

- Integración completa en la Web.
- Potente e intuitiva consulta de disponibilidad; con calendarios, indicador de estado, formulario de búsqueda general e individual, con posibilidad de filtrar los resultados por tipo de alojamiento, ubicación, capacidad y habitaciones.
- Acceso al calendario completo de disponibilidad de cada alojamiento (general o individual).
- Planificador de disponibilidad con visualización de estado por colores, con filtro de búsqueda y movilidad sobre él por meses y años (adelante y hacia atrás).
- Ayuda y orientación de cómo efectuar una reserva. Acceso a las condiciones generales de la reserva, cláusulas de cancelación y preguntas frecuentes.
- Visualización del precio desglosado. Pago seguro, contratación según la Ley de Servicios de la Sociedad de la Información (L.S.S.I.). Confidencialidad según LOPD.
- Recepción inmediata de las solicitudes mediante correo electrónico (con copia para la empresa y copia para el usuario).
- Una vez verificado el cobro, las reservas se visualizan inmediatamente en pantalla, se registran on-line, se depositan en una ‘Bandeja de entrada’ y se actualiza la disponibilidad. Las reservas se gestionan en un formato que permita su integración en cualquier otro sistema de gestión externo.
- Fidelización de clientes.

Evidentemente, para representar el comportamiento de un sistema como éste –y en el ámbito del análisis del dominio—, se debería investigar los usos y prácticas comunes al área de negocio que faciliten la comprensión del funcionamiento del sistema y de las necesidades del cliente. Falta información adicional como, por ejemplo, cómo se tratan las tarifas en relación a las temporadas turísticas o a los tipos, categorías o capacidad de las habitaciones (doble –con dos camas o una—, especial o junior suite, superior o suite, etc.) o al régimen de alojamiento (sólo alojamiento, con desayuno, media pensión o pensión completa).

Piense que, al hacer el análisis de este sistema, tendrá que hacer un modelo que represente cómo se comporta la totalidad del sistema; en cualquier situación y para todos los usuarios potenciales.

- a) Desarrolle qué información debe obtener de su cliente en las sucesivas entrevistas que mantendrá con él. (Por ejemplo, ¿cómo se gestionan los cupos de reservas de las agencias? o ¿se puede hacer una reserva con habitaciones de distintas categorías? (3 puntos)**
- b) Con la información del enunciado y las conclusiones del punto a), construya un listado –lo más completo y estructurado posible— de los requisitos o especificaciones del producto. (2 puntos)**

Solución

Lo que sigue a continuación NO son las respuestas que se exigirían al ejercicio, sino que se ha desarrollado de una manera más detallada con la intención de que ilustre al alumno en relación a las reflexiones, deducciones y soluciones que se deben aportar durante el proceso de Análisis de una aplicación. Se complementa con el documento '[Análisis Reserva Online.pdf](#)'. JOSÉ FÉLIX ESTÍVARIZ

- a) El objetivo principal de la fase de Análisis es conocer las necesidades del Cliente y entender correctamente QUÉ tiene que hacer la aplicación.

Una vez determinado que los destinatarios de la aplicación (el Cliente) son hoteles particulares (posiblemente en competencia unos con otros), habrá que averiguar en qué consiste la parcela del área de negocio en la que va a utilizarse el producto, qué actividad realiza el Cliente y así potenciar las ventajas que el sistema pueda tener para su negocio. De esta forma, se debería conocer:

- Si el hotel tiene Web propia.
- Qué servicios se ofrecen en la Web.
- Qué servicios ofrece el hotel a sus clientes en general (los de la Web pueden ser sólo una parte).
- Si tienen acuerdos con agencias de turismo, cupos, o tratamientos especiales con empresas o instituciones.
- Cómo gestiona, hasta ahora, las reservas, la disponibilidad y la ocupación de los servicios del hotel (¿de manera no informatizada?, ¿con una aplicación de gestión ya implantado?, ¿con otra aplicación de reservas online?)
- Si se gestionaba la ocupación de una manera mixta o no informatizada, ¿está dispuesto a cambiar sus procedimientos de trabajo para realizar una gestión informatizada y unificada?
- ¿De qué manera se hace el mantenimiento de la Web del hotel?

Alrededor del 75% del alojamiento hotelero se contrata a través de un sistema online de este tipo. Fundamentalmente consiste en posibilitar, al usuario interesado en el hotel, hacer búsquedas de disponibilidad de servicios y contratarlos por Internet. Esta descripción correspondería a la funcionalidad de los usuarios externos al hotel: la vista 'Operativa de Reservas'. Pero debe haber otra parte que permita la gestión del producto por parte del hotel: la vista de 'Gestión del Sistema'.

Para cualquiera de las dos vistas, el producto tendrá una parte importante invariante y válida para todos los hoteles. Otra parte será la personalización para cada hotel. En ningún caso, el desarrollo de esta parte y el ajuste de la personalización deben suponer una carga adicional de trabajo para el hotel.

Para esta parte de personalización, es necesario conocer:

- Una catalogación de los servicios del hotel, clasificados por categorías, subcategorías, tipos, características e instancias; con la denominación que utiliza el hotel para cada uno de los elementos anteriores. De esta manera, la denominación de un servicio (por ejemplo una categoría determinada de habitación, que en el hotel se denomina '*junior suite*') aparecerá personalizado.
- La mayoría de los servicios serán visibles para la reserva online; pero puede que el hotel no quiera que otros aparezcan. ¿Cuáles sí y cuáles no?
- Cada 'tipo' de servicio tiene asociado un calendario de tarifas. Hay ofertas que son modificaciones del calendario de tarifas. Sin embargo, los paquetes especiales y las promociones son ligaduras entre dos o más servicios a los que se subordina un cambio de

tarifas. Por ejemplo, unas ‘jornadas gastronómicas’ pueden incluir el servicio de alojamiento y el de restaurante durante un período de tiempo determinado y a un precio especial. Una promoción de este tipo ¿se ofrece como un servicio más o se muestra como una variación de la tarifa de alojamiento ligada al uso obligatorio del restaurante?

- ¿Qué período máximo y mínimo se permite en las búsquedas? Para un período de tiempo ¿se permite buscar y reservar subcategorías, tipos o incluso servicios diferentes? Por ejemplo, una habitación del hotel y una de apartamento o toda una planta del hotel, dos salas de reunión y el restaurante para un congreso.
- ¿Qué condiciones de cancelación tiene el hotel? ¿Son únicas o diferenciadas para los distintos servicios? ¿Las promociones tienen condiciones de cancelación especiales?
- ¿Tiene contratado el Cliente alguna cuenta para servicio de Pago Seguro (por ejemplo PayPal, BMT Micro, etc.)?
- En el transcurso del funcionamiento de la aplicación, la información que se obtiene del hotel se traduce a un formato interno del sistema de reservas. Es esa ‘copia’ de los datos del hotel con la que funciona el sistema y el formato interno de representación es único y válido para cualquier hotel. Dicho formato será el que dé la máxima compatibilidad y funcionalidad posible. El usuario visualizará la información de la ‘copia’ que maneja el sistema, pero en los términos y con la funcionalidad que utilice y decida el hotel.

En general, habrá que averiguar todo lo necesario para poder traducir la información que el hotel ofrezca a los usuarios (y el tratamiento que el Cliente quiera darle) al formato interno y mecánica de funcionamiento de la parte invariante del producto.

En cuanto a la vista de ‘Gestión del Sistema’, la información que hay que obtener del Cliente se refiere a:

- ¿Cómo se gestiona internamente la ocupación del hotel (en lo que se refiere a los servicios susceptibles de reserva)?
- Cuando el hotel lanza un nuevo producto o promoción ¿cómo lo ofrece al público? ¿cómo se ofrece a las agencias? Es posible que el hotel publique una promoción en su Web pero considere que no es adecuado incluirla en el sistema de reservas ¿es factible que, quien mantenga la Web del hotel, marque los contenidos que publique con un atributo de disponibilidad para los usuarios de la reserva?
- ¿Qué tratamiento se les da a las agencias y sus cupos, a los clientes especiales, empresas e instituciones?
- ¿De qué manera quiere visualizar el calendario de ocupación y qué información es relevante para hacer búsquedas en él?
- ¿Tiene programas de fidelización de sus clientes? ¿Quiere gestionarlos con este producto?
- ¿Es socio de la Agencia Española de Protección de Datos? ¿Tiene intención de inscribirse? Los aparatos que van a tener acceso a la aplicación de reservas ¿tienen un control de la identidad del usuario? El tratamiento actual de los datos privados de los clientes del hotel ¿tiene garantías para preservar su confidencialidad?

- b) La aplicación estaría integrada en Web y tendría dos vistas: la de los usuarios externos al hotel sería la vista de la 'Operativa de Reservas'; la del personal autorizado del hotel sería la vista de 'Gestión del Sistema'.

Los diagramas preliminares DFD de cada vista podrían ser estos:

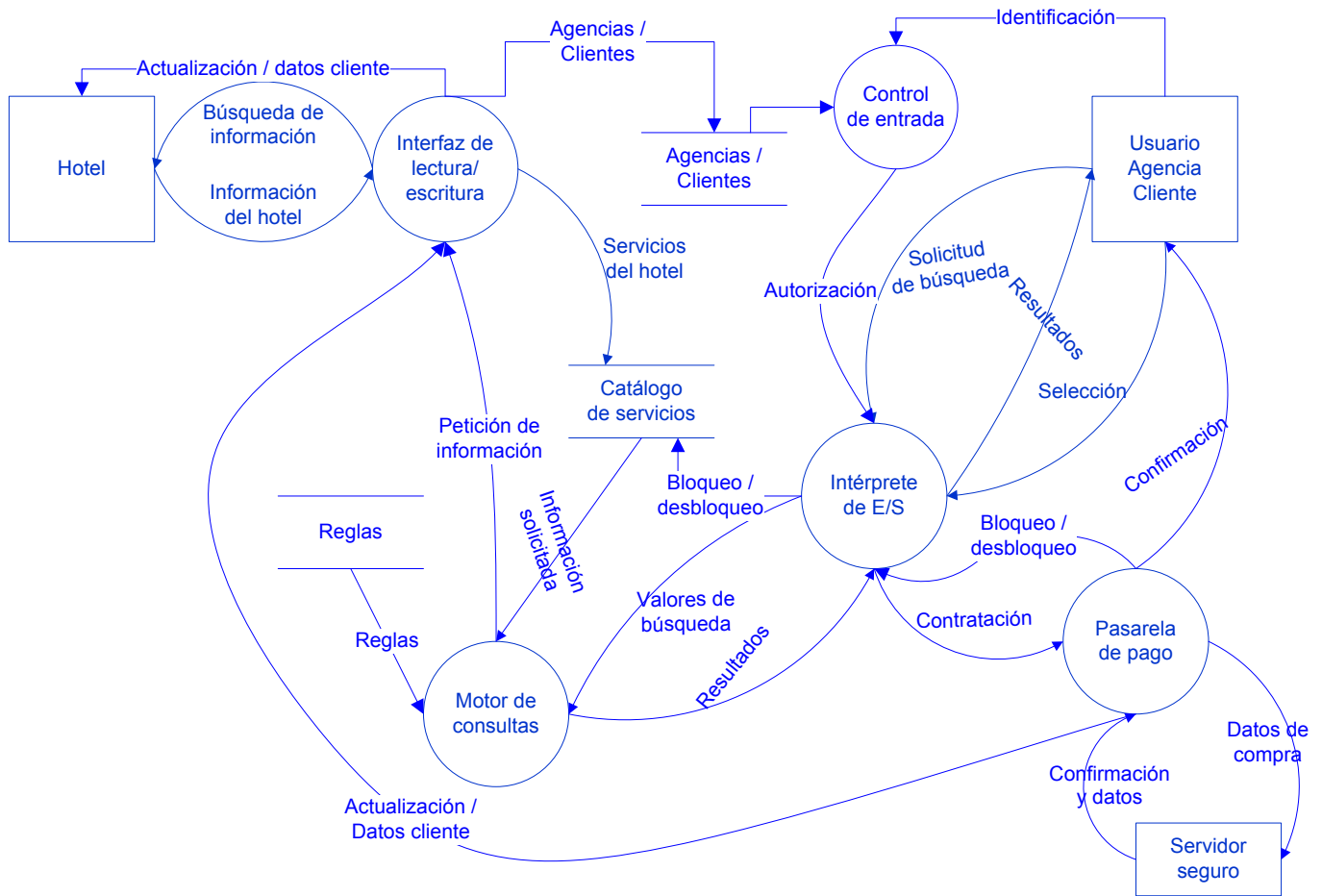


Diagrama 1 DFD de la vista de la 'Operativa de Reservas'

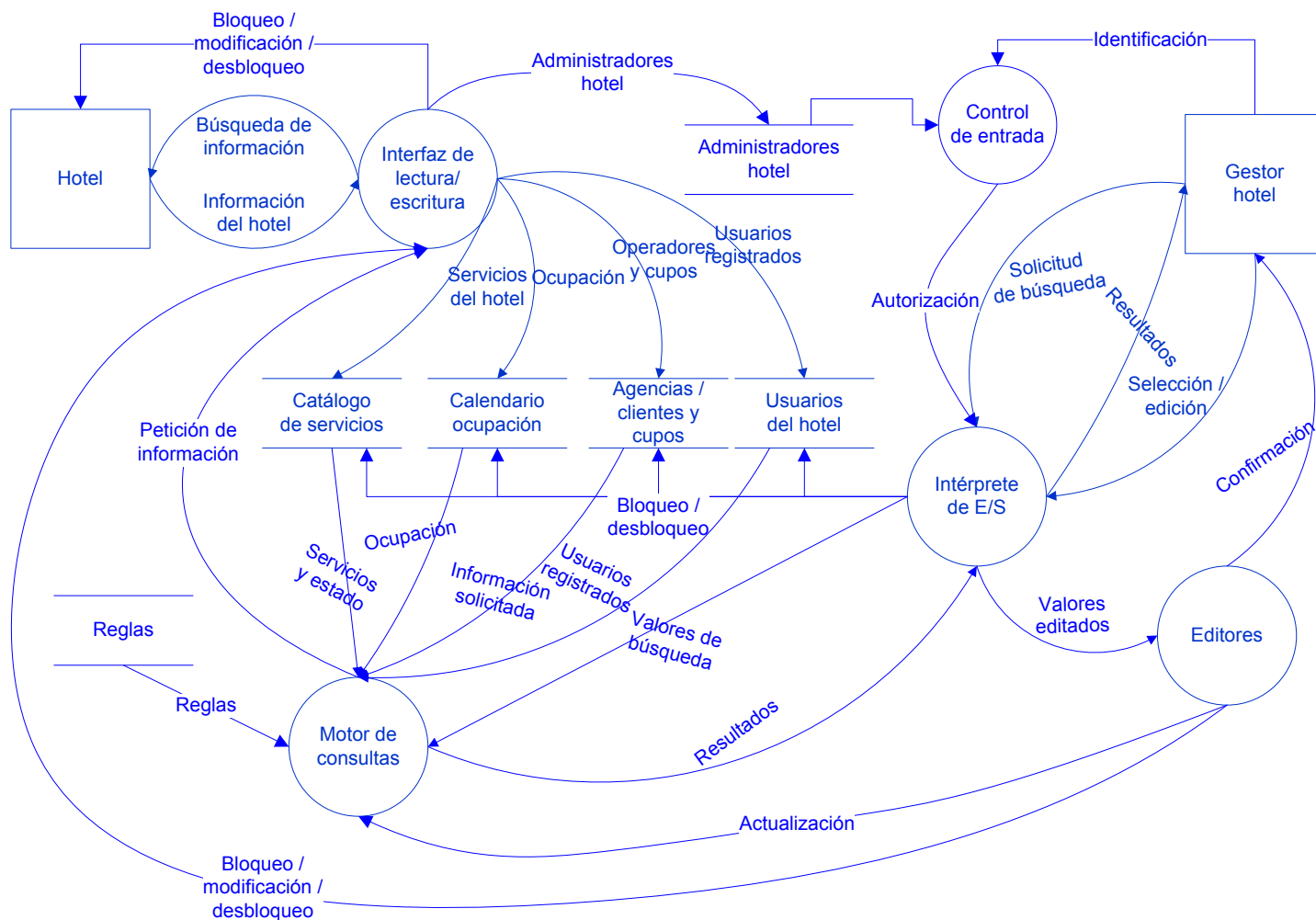


Diagrama 2 DFD de la Vista de Gestión

En cuanto a los requisitos de la aplicación:

- Aplicación integrada en Web. Tendría dos vistas: la de los usuarios externos al hotel sería la vista de la ‘Operativa de Reservas’; la del personal autorizado del hotel sería la vista de ‘Gestión del Sistema’.
- Para cualquiera de las dos vistas, el producto tendrá una parte importante invariante y válida para todos los hoteles. Otra parte será la personalización para cada hotel.
- En ningún caso, el desarrollo de la parte específica del hotel y el ajuste de la personalización deben suponer una carga adicional de trabajo para el hotel.
- La aplicación trabajará con una representación interna de la información del hotel. Sólo modificará los datos internos del hotel para registrar una reserva confirmada, para actualizar el calendario de ocupación o desde la vista de ‘Gestión del Sistema’ por un usuario autorizado por el hotel.
- El ‘calendario de ocupación’ será único. Tanto éste como el resto de la información del hotel que se maneja, es propiedad del hotel y se accederá a dicha información según el formato que establezca el Cliente.


- La ‘personalización’ consistirá en que, tanto la funcionalidad como la visualización de la aplicación, se desarrollará según los procedimientos de trabajo establecidos por el hotel para su negocio.

Funcionalidad para la ‘Operativa de Reservas’:

- Búsqueda de disponibilidad y selección de alojamientos disponibles con unas características determinadas –número de habitaciones, tipos de habitaciones, período de tiempo, características de los usuarios de cada habitación, régimen de alojamiento, etc.-
- Potente e intuitiva consulta de disponibilidad; con calendarios, indicador de estado, formulario de búsqueda general e individual, con posibilidad de filtrar los resultados por tipo de alojamiento, ubicación, capacidad y habitaciones.
- Ayuda y orientación de cómo efectuar una reserva. Acceso a las condiciones generales de la reserva, cláusulas de cancelación y preguntas frecuentes.
- Aplicación de tarifas, ofertas y paquetes promocionales del hotel o acordados con una empresa determinada.
- Aceptación de la selección y redirección para realizar la contratación en condiciones de seguridad. Si no se acepta, se puede volver a hacer la búsqueda y hacer la selección con otras condiciones.
- Visualización del precio desglosado. Pago seguro, contratación según la Ley de Servicios de la Sociedad de la Información (L.S.S.I.). Confidencialidad según LOPD.
- Recepción inmediata de las solicitudes mediante correo electrónico (con copia para la empresa y copia para el usuario).

Funcionalidad para la ‘Gestión del Sistema’:

- Dar de alta, baja o modificar un usuario con privilegios de administración sobre la aplicación.
- Modificar el catálogo de servicios –no su estructura—, añadir un servicio, eliminarlo, modificar sus atributos, marcarlo como disponible para algún tipo de usuario externo o lo contrario.
- Potente e intuitiva consulta de disponibilidad; con calendarios, indicador de estado, formulario de búsqueda general e individual, con posibilidad de filtrar los resultados por tipo de alojamiento, ubicación, capacidad y habitaciones.
- Acceso al calendario completo de disponibilidad de cada alojamiento (general o individual).
- Modificar el ‘Calendario de ocupación’, bien por categorías de servicios, de un servicio particular o de todo el catálogo del hotel.
- Dar de alta, baja o modificar los datos de agencias y empresas clientes, asignarles o modificarles cupos y servicios.
- Añadir, eliminar y modificar los datos de los usuarios de los servicios del hotel. Incluirles en promociones, programas de fidelización, etc.
- Realizar planificaciones de servicios, de un servicio particular o de todo el catálogo del hotel, con visualización de estado por colores, con filtro de búsqueda a través de ventanas temporales de días, semanas, meses o años y movilidad adelante y hacia atrás.
- Una vez verificado el cobro, las reservas se visualizan inmediatamente en pantalla, se registran on-line, se depositan en una ‘Bandeja de entrada’ y se actualiza la disponibilidad. Las reservas se gestionan en un formato que permita su integración en cualquier otro sistema de gestión externo.
- Fidelización de clientes.

INGENIERÍA DEL SOFTWARE (2º Curso) Cód. 53210 SISTEMAS 54208 GESTIÓN				
	Modelo			
	Septiembre 2009			
				Ámbito
				NACIONAL - UE RESERVA

ESTE EJERCICIO NO ES DE TEST. NO TIENE RETORNO TELEMÁTICO.

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. Describa las notaciones que conozca para la especificación de requisitos de un sistema software.

Solución

Página 49 del texto básico de la asignatura.

Lenguaje natural, diagramas de flujo de datos, diagramas de transición de estados y pseudocódigo. Para el modelo de datos se emplean notaciones como el diccionario de datos o el diagrama entidad-relación.

2. Conteste a las siguientes preguntas:
 - a) ¿El uso de pruebas garantiza la ausencia de errores en un programa?
 - b) ¿Cómo puede estimarse la cantidad de errores que un juego de pruebas no es capaz de detectar en un módulo?

Solución

- a) Generalmente, probar exhaustivamente un programa es inabordable y, además, no resulta ni rentable ni práctico. Por lo tanto, las pruebas de software, frente a la verificación formal, no garantizan la ausencia total de errores.
- b) Mediante el siguiente procedimiento (descrito en la sección “5.7.4. Estimación de errores no detectados”):
 1. Se anota el número de errores **EI** que el juego de pruebas produce en el módulo.
 2. Se corrige el módulo hasta que el juego de pruebas deje de producir errores.
 3. Se introducen aleatoriamente en el módulo cierto número de errores **EA**.

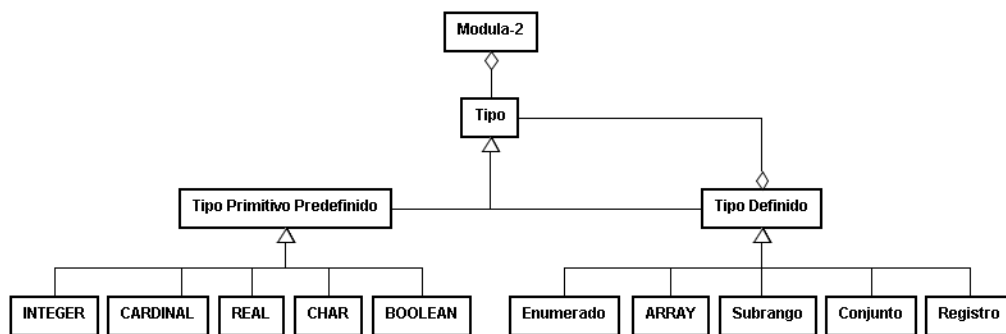
4. Se vuelve a pasar el juego de pruebas sobre el módulo, con lo que se producirán **ED** errores.
5. El número estimado de errores sin detectar será: **EE = (EA - ED)*(EI / ED)**.


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

3. **Modele la siguiente especificación con un Diagrama de Objetos:**

“El lenguaje de programación Modula-2 dispone de los siguientes tipos primitivos predefinidos: INTEGER, CARDINAL, REAL, CHAR y BOOLEAN. Además, soporta la definición por parte del programador de nuevos tipos Enumerado, Subrango, ARRAY, Conjunto y Registro. Los nuevos tipos se crean a partir de los tipos primitivos y/o cualquier otro tipo definido previamente por un programador.”

Solución



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.

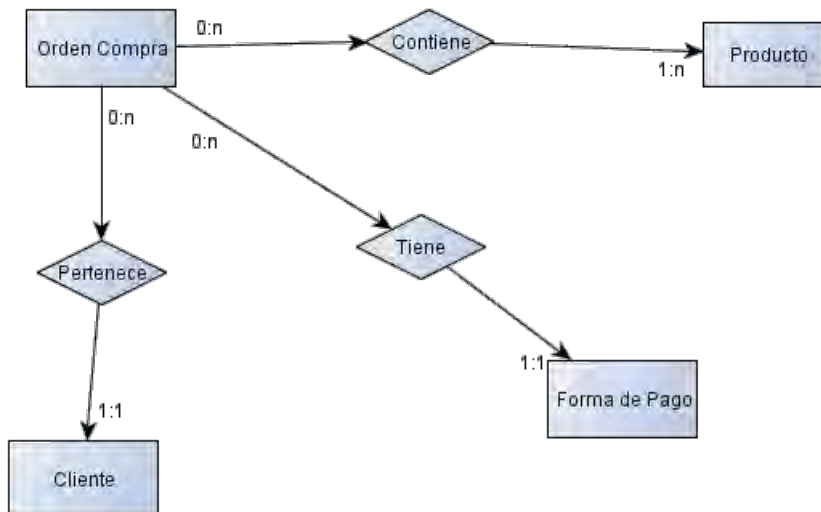
En ocasiones es necesaria para el mantenimiento de dichos productos software. (Pág. 26)

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

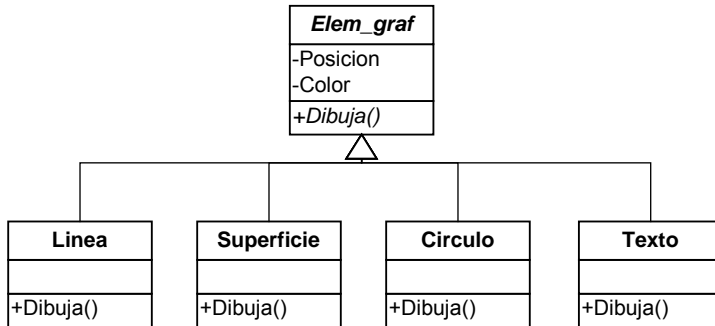
Usuario = {CaracterAlafnumérico}¹⁰

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 líneas, cadenas de texto, círculos, superficies, 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 línea o con texto.



Un escalón adicional en el diseño es comprender que la aplicación cliente querrá manejar dibujos (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 línea o círculo. 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 `Contenedores` 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 `Dibujos`, 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**.

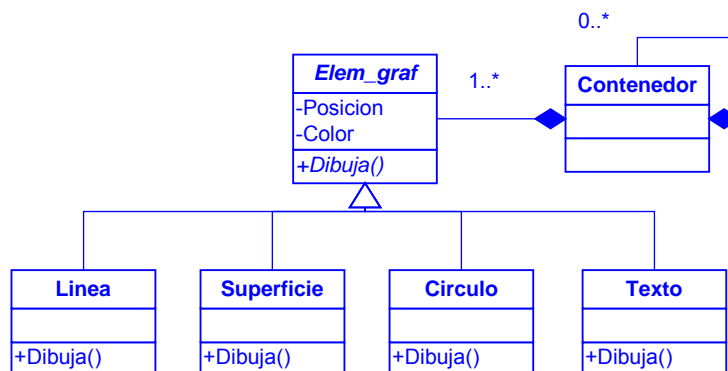
Supóngase ahora un entorno gráfico de ventanas como Windows donde los elementos que se manejan (Objeto_entorno) son del tipo Dibujo. También aquí hay una serie de elementos simples (como pueden ser Borde, Icono, Fondo, Etiqueta o Decorador) y otros Elem_compuesto son compuestos, como Ventana, Cuadro, 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 desplegables, por Separadores y un Asa (estos últimos, los consideramos especializaciones de Decorador). Por ejemplo:



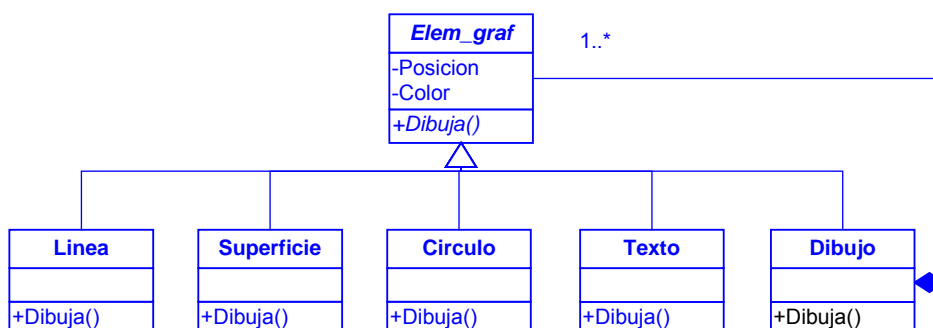
Pregunta c) Utilice el patrón de diseño ‘Composición’ anterior para representar la estructura de Objeto_entorno y, en concreto, Barra_herr.

Solución

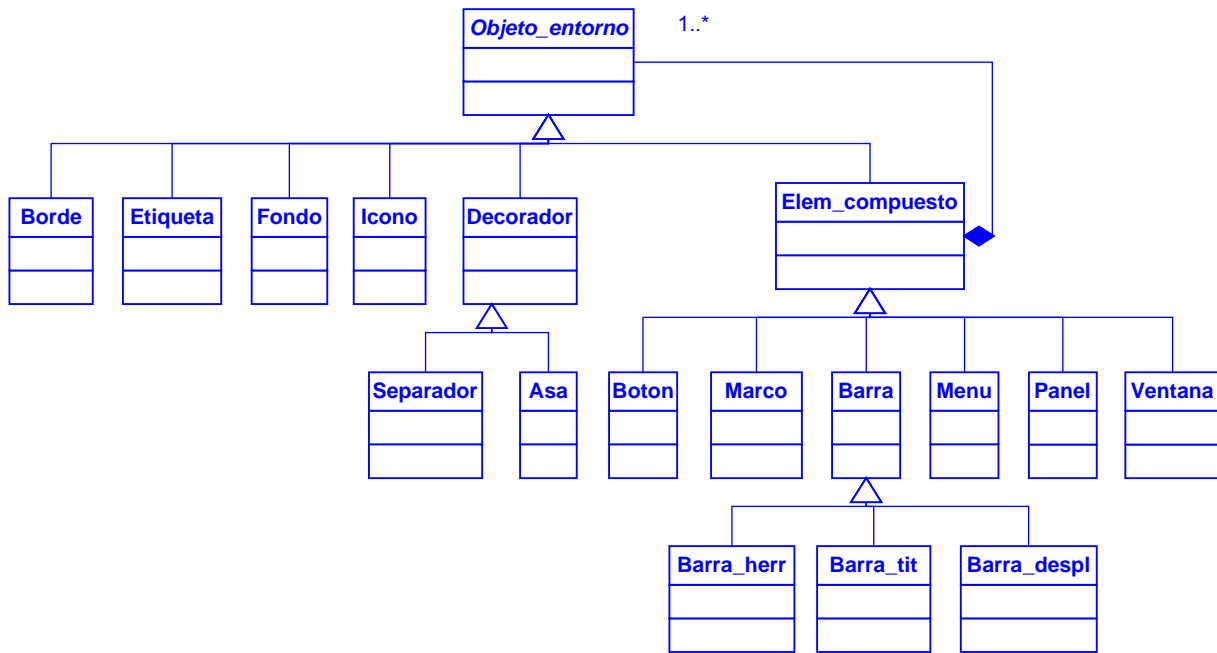
a) Clase Contenedor:




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 'luevan' 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

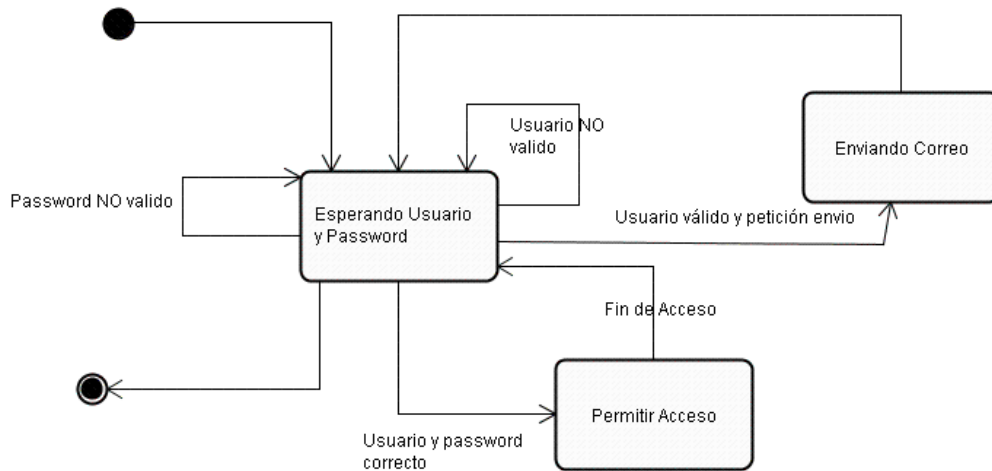
decir que sea el único. Lo más común es que haya que considerar necesidades y aspectos relacionados con el futuro uso de la aplicación o tener en cuenta decisiones estratégicas al respecto de la competitividad del producto en el mercado o de la propia organización. Esto contribuye a la volatilidad de 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 petionario 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:

- 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.
- Describalo 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
  
```

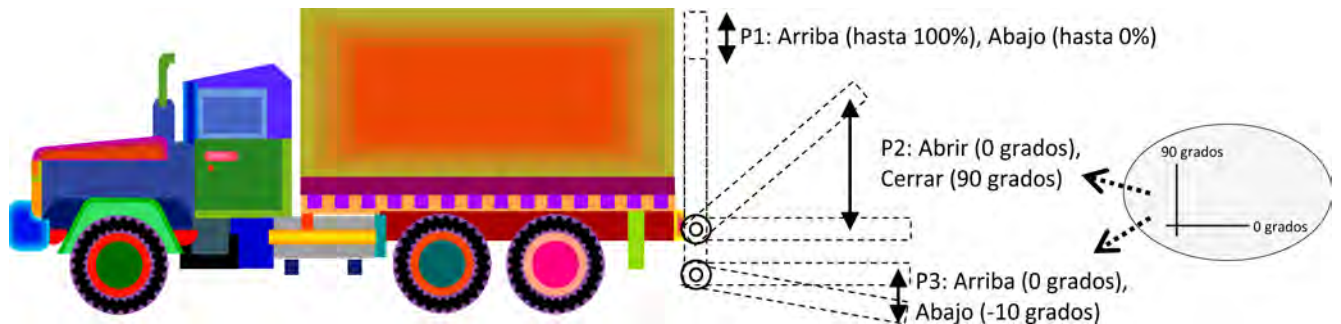
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

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)

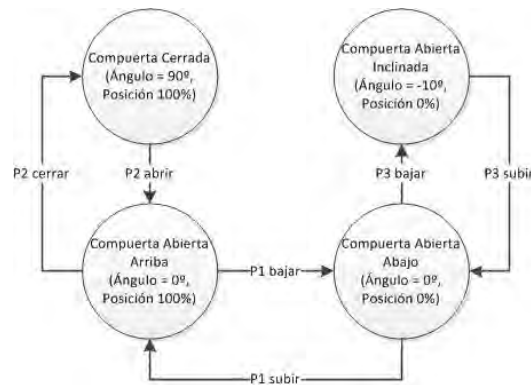
El sistema informático deberá asegurar que la compuerta sólo se abre o se cierra si está arriba del todo (100% de la posición vertical).



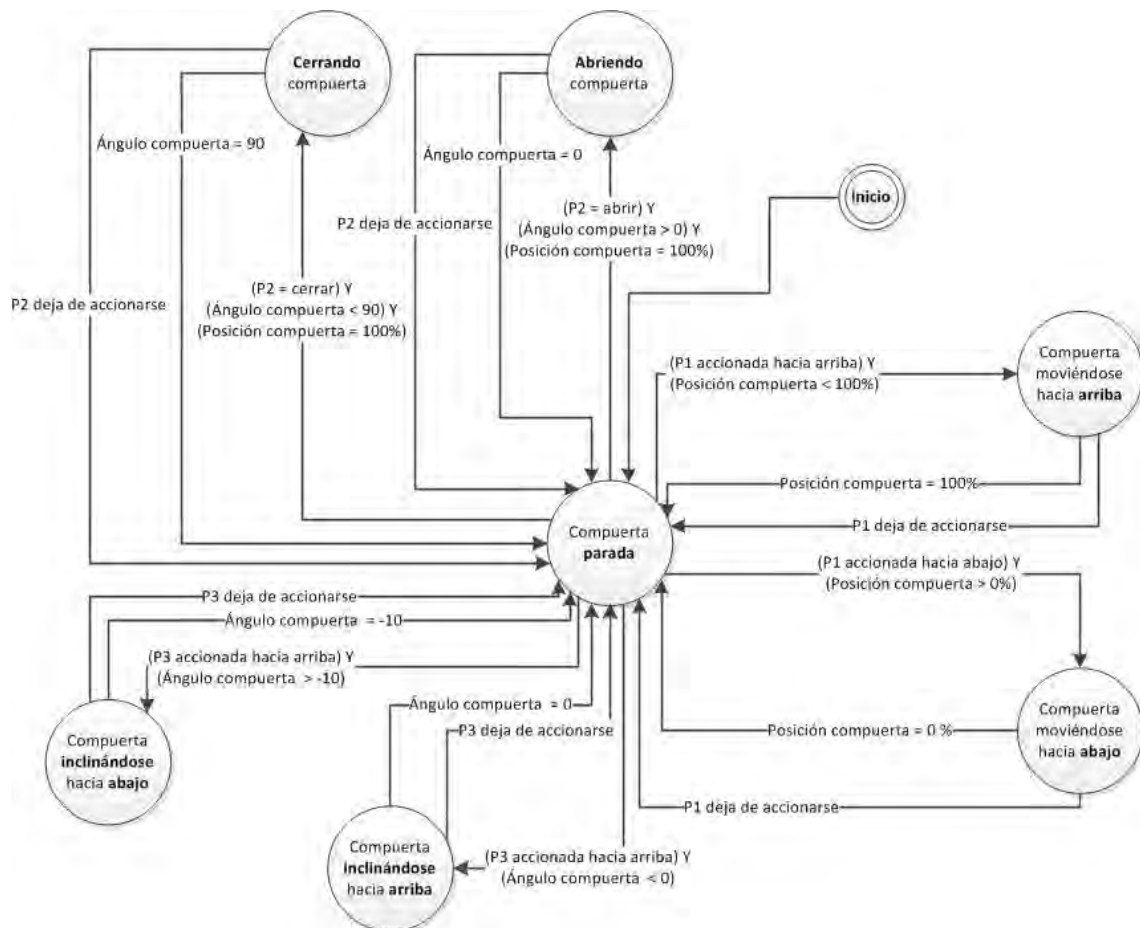
Diseñe el sistema informático mediante un diagrama de transición de estados. No contemple la posibilidad de accionar más de una palanca a la vez.


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)

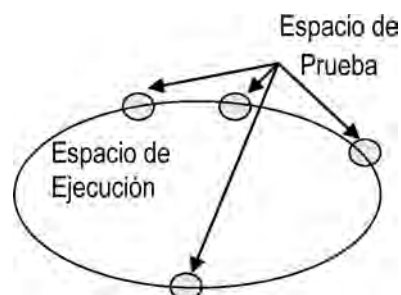
PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

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.

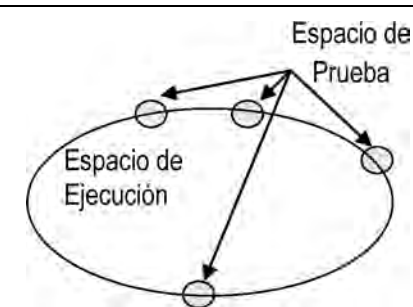
b)

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.”

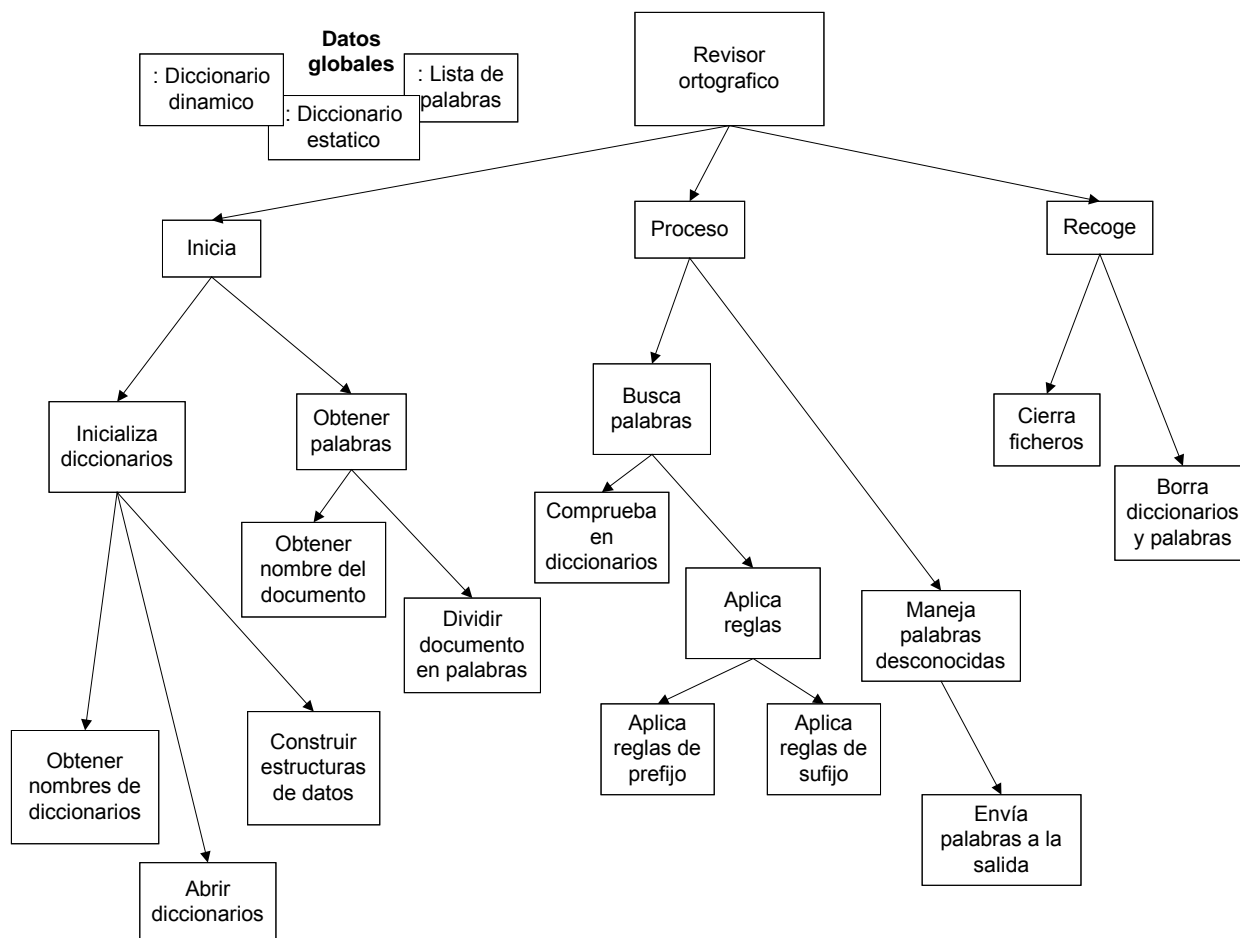
Las especificaciones y objetivos iniciales del programa son:

- 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:

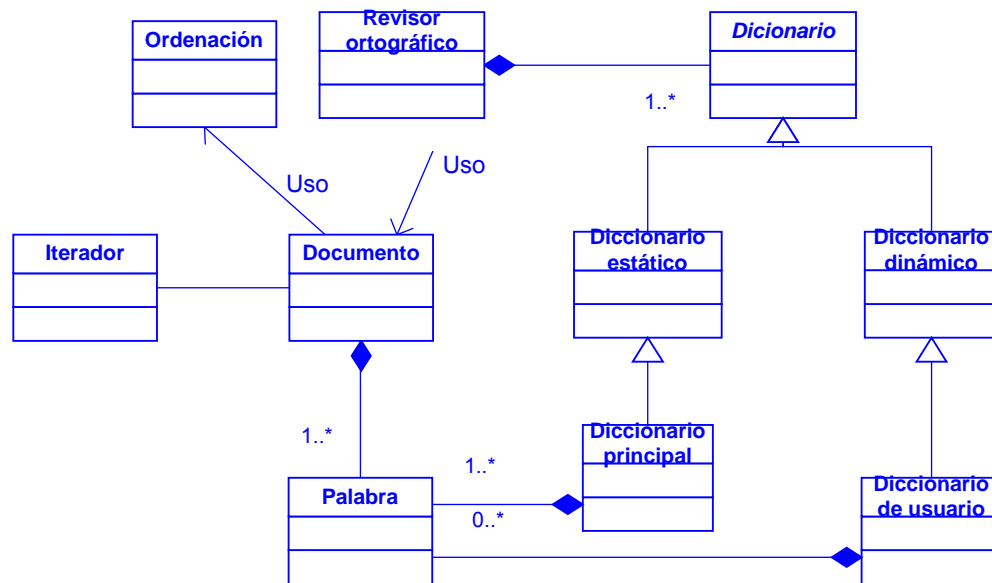



Las desventajas de este diseño son:

- 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



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.

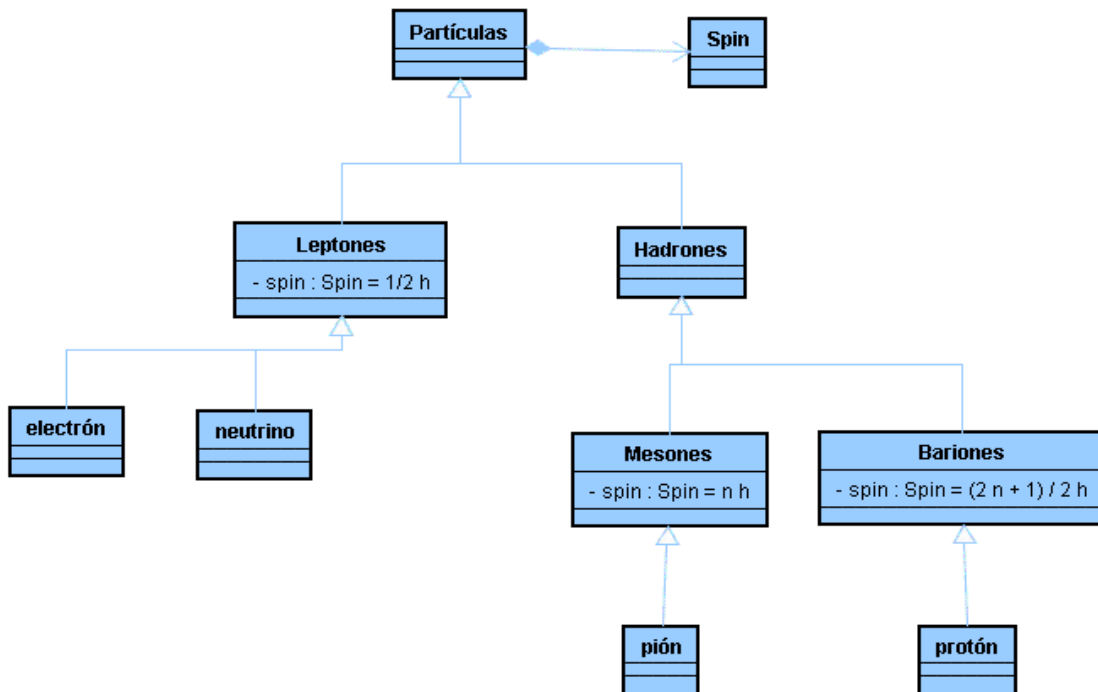
SEGUNDA PARTE. PREGUNTA DE TEORÍA APLICADA (MÁXIMO 5 PUNTOS)

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



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)

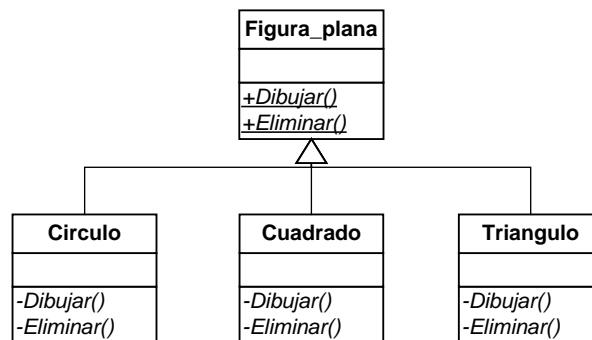
PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. Defina el concepto de '*configuración software*'. Explique qué es y cómo se utiliza la '*línea base*' para gestionar los cambios en la configuración del software.

Solución

Epígrafe 1.9.5 del libro; páginas 30 a 32.

2. En el diagrama de objetos de la figura, observe que **Figura_plana** es una clase con polimorfismo. El método **Circulo.Dibujar()** sobrecarga el método de **Figura_plana**; y ocurre lo mismo con **Eliminar()** y con los respectivos métodos de **Cuadrado** y **Triangulo**.



El operador **new** se encarga de crear una instancia invocando a un método –denominado *constructor*– de una clase o un objeto determinado. Así, la sentencia:

Figura_plana Mi_figura = **new** **Circulo()**

crea la instancia **Mi_figura**, de tipo **Figura_plana**, mediante el constructor de **Circulo** –que es el método **Circulo()**, implícito en la clase **Circulo**–.

- a. ¿La sentencia anterior generaría correctamente el código o el compilador daría un mensaje de error por diferencia en los tipos? ¿Qué tipo se asigna, finalmente, a **Mi_figura**?
- b. Si se hace la llamada **Mi_figura.Dibujar()**, ¿qué método se ejecuta, el de **Figura_plana** o el de **Circulo**?

Solución

El hecho de *conectar* la invocación de un método con el código que le corresponde, se llama **acoplamiento** o **enlace**. Cuando el compilador conoce exactamente qué código le corresponde a la invocación de un método, la asignación se denomina *enlazado temprano* (*early binding*) y se produce durante la compilación, antes de la ejecución. Sin embargo, si durante la compilación no se sabe qué código le corresponde exactamente en la invocación de un método, la asignación se resuelve durante la ejecución del programa, cuando las referencias toman un valor concreto. A esto se le llama *acoplamiento tardío* (*late binding*) o **enlazado dinámico**.

- a. En el caso de la sentencia anterior, se crea un objeto de tipo **Circulo** e, inmediatamente, la referencia obtenida se asigna a un objeto, `Mi_figura`, de tipo **Figura_plana**. La sentencia es correcta y el compilador no daría error puesto que, por la herencia, un **Circulo** *es*, también, una **Figura_plana**. A esto se le llama **generalización** y es una transformación del tipo hacia arriba (*upcasting*).
- b. Se ha establecido que el tipo que el compilador ha asignado a `Mi_figura` es **Figura_plana** -enlazado temprano—. Sin embargo al invocar, **durante la ejecución**, al método `Mi_figura.Dibujar()`, el **polimorfismo** -o enlazado dinámico— garantiza que el método que se ejecuta en la invocación es la especialización correcta de **Circulo** -`Circulo.Dibujar()`— puesto que ese es el tipo de `Mi_figura`.

En las descripciones anteriores se pone de manifiesto de qué manera, la herencia y el polimorfismo, permiten diseñar con ocultación, ahorrar en el desarrollo de código -el cual resulta compacto, sencillo y fácilmente entendible— y lleva a diseños flexibles en los que es fácil añadir o eliminar elementos.

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, articulada, de un camión. La compuerta está formada por dos plataformas cuyo movimiento se realiza con dos motores –M1 y M2— accionados mediante tres palancas –P1, P2 y P3—. Unos sensores, llamados ‘*final de carrera*’ detectan que cada plataforma ha llegado al límite de su movimiento: FC1_M1, FC2_M1, FC1_M2, FC2_M2 y FC3_M2. Una vez que se activa el sistema, las órdenes de las palancas se envían a un pequeño autómatas que alimenta a los motores. Dicho autómatas debe contener la lógica necesaria para que las maniobras de Apertura y Cierre sean seguras y correctas. Según la figura de más abajo:

- 1) La palanca P1 controla M1 y permite subir y bajar la compuerta en vertical. El sensor FC2_M1 indicará si la compuerta está a ras del suelo (0%, FC2_M1=1) y FC1_M1 si está arriba del todo (100%, FC1_M1=1).
- 2) La palanca P2 acciona el M2 para abrir y cerrar la compuerta –la segunda plataforma—. Los finales de carrera indicarán el ángulo correcto de la apertura de la compuerta. Se considera que la compuerta está abierta si el ángulo es de 0 grados (FC2_M2=1) y cerrada si el ángulo es de 90 grados (FC1_M2=1).
- 3) La palanca P3 acciona también el M2 e inclina más la plataforma para facilitar la carga y descarga del camión. El ángulo que puede girarse la compuerta con esta palanca va de 0 (FC2_M2=1) a -10 grados (FC3_M2=1).

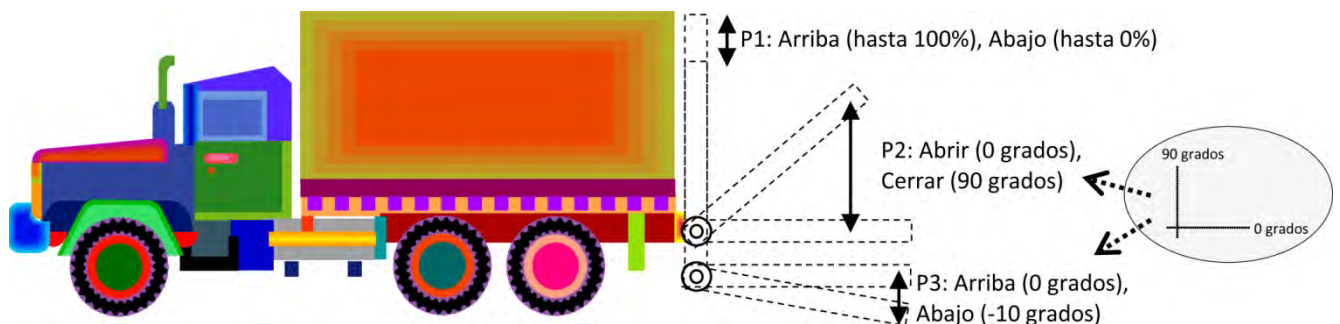
Las maniobras de Apertura y Cierre son las siguientes:

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)

Restricciones:

- No se pueden pulsar dos palancas a la vez.
- La plataforma **NO** se inclinará –en ningún sentido— si no está **totalmente abierta** (ángulo ≤ 0 grados) y, además, la **compuerta totalmente bajada** (0% ó FC2_M1=1)
- La plataforma **NO** se cerrará si está **inclinada** (ángulo ≤ 0 grados) o la **compuerta no está totalmente subida** (elevación $< 100\%$ ó FC1_M1 $\neq 1$)
- La compuerta **NO** se elevará ni bajará hasta que la **plataforma esté totalmente horizontal** (ángulo = 0 grados o FC2_M2=1)



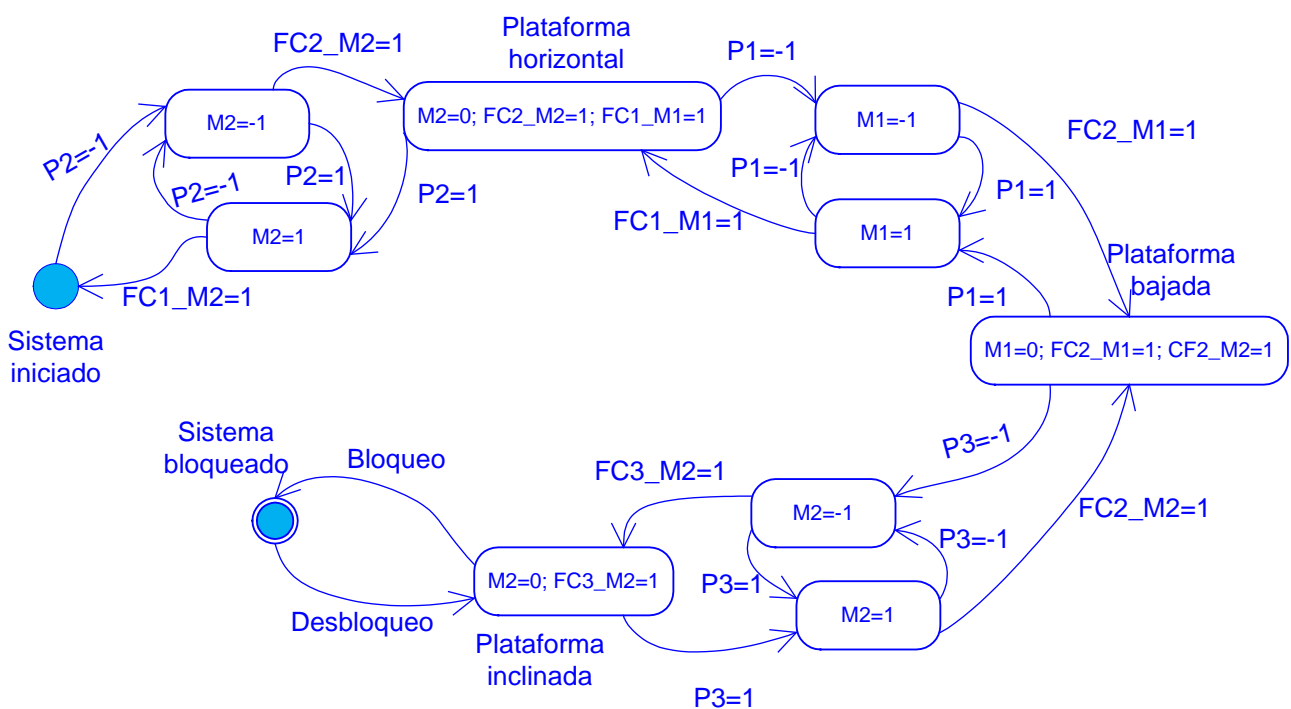
Se pide diseñar el funcionamiento descrito mediante un diagrama de transición de estados.

Solución

Supongamos que los valores que pueden tomar las variables del sistema son las siguientes:

Motores		Finales de carrera		Palancas	
M1	$=\{-1, 0, 1\}$ Bajar, parado, subir	FC1_M1	$=\{0, 1\}$	P1	$=\{-1, 0, 1\}$ Abajo, no pulsado, arriba
		FC2_M1	$=\{0, 1\}$		
M2	$=\{-1, 0, 1\}$ Abrir o bajar, parado, cerrar o subir	FC1_M2	$=\{0, 1\}$	P2	$=\{-1, 0, 1\}$ Abrir, no pulsado, cerrar
		FC2_M2	$=\{0, 1\}$		
		FC3_M2	$=\{0, 1\}$	P3	$=\{-1, 0, 1\}$ Abajo, no pulsado, arriba

Las entradas al sistema serán las acciones de las palancas y el valor de los finales de carrera; las salidas, las señales de los motores. Los estados vendrán definidos por el movimiento de los motores y las posiciones de la compuerta. El diagrama de estados sería:



En el estado inicial, se verifican y calibran los accionadores, motores y finales de carrera. Los valores internos deben ser (estado 0):

ESTADO 0, REPOSO	
MOTORES	FINALES DE CARRERA
M1 =0	FC1_M1 =1
	FC2_M1 =0
M2 =0	FC1_M2 =1
	FC2_M2 =0
	FC3_M2 =0

La acción de la palanca P2 para abrir, permite la transición al movimiento de M2 hasta poner la plataforma horizontal o hasta que se pulse P2 en sentido contrario (1):

ESTADO 1, ABRIENDO	
MOTORES	FINALES DE CARRERA
M1 =0	FC1_M1 =1
	FC2_M1 =0
M2 =-1	FC1_M2 =0
	FC2_M2 =0
	FC3_M2 =0

ESTADO 2, ABIERTO	
MOTORES	FINALES DE CARRERA
M1 =0	FC1_M1 =1
	FC2_M1 =0
M2 =0	FC1_M2 =0
	FC2_M2 =1
	FC3_M2 =0

Al llegar a la posición horizontal el motor M2 se detiene y el sistema queda en el siguiente estado (2, de reposo).

A continuación, la sucesión del resto de estados:

ESTADO 3, BAJANDO	
MOTORES	FINALES DE CARRERA
M1 =-1	FC1_M1 =0
	FC2_M1 =0
M2 =0	FC1_M2 =0
	FC2_M2 =1
	FC3_M2 =0

ESTADO 6, INCLINADO	
MOTORES	FINALES DE CARRERA
M1 =0	FC1_M1 =0
	FC2_M1 =1
M2 =0	FC1_M2 =0
	FC2_M2 =0
	FC3_M2 =1

ESTADO 9, SUBIR	
MOTORES	FINALES DE CARRERA
M1 =1	FC1_M1 =0
	FC2_M1 =0
M2 =0	FC1_M2 =0
	FC2_M2 =1
	FC3_M2 =0

ESTADO 4, ABAJO	
MOTORES	FINALES DE CARRERA
M1 =0	FC1_M1 =0
	FC2_M1 =1
M2 =0	FC1_M2 =0
	FC2_M2 =1
	FC3_M2 =0

ESTADO 7, BLOQUEO	
MOTORES	FINALES DE CARRERA
M1 =0	FC1_M1 =0
	FC2_M1 =1
M2 =0	FC1_M2 =0
	FC2_M2 =0
	FC3_M2 =1

ESTADO 10, CERRANDO	
MOTORES	FINALES DE CARRERA
M1 =0	FC1_M1 =1
	FC2_M1 =0
M2 =1	FC1_M2 =0
	FC2_M2 =0
	FC3_M2 =0

ESTADO 5, INCLINANDO	
MOTORES	FINALES DE CARRERA
M1 =0	FC1_M1 =0
	FC2_M1 =1
M2 =-1	FC1_M2 =0
	FC2_M2 =0
	FC3_M2 =0

ESTADO 8, ARRIBA	
MOTORES	FINALES DE CARRERA
M1 =0	FC1_M1 =0
	FC2_M1 =1
M2 =1	FC1_M2 =0
	FC2_M2 =0
	FC3_M2 =0

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. Supóngase que una organización decide afrontar, por primera vez, un proyecto software en el que no tiene experiencia y que se sitúa en un ámbito desconocido para ella. Desde el punto de vista del ciclo de vida, indique qué alternativas tiene esta organización para culminar el proyecto con éxito y explique cómo pueden, dichas alternativas, mitigar los problemas potenciales con los que la organización se puede encontrar durante el desarrollo.

Solución

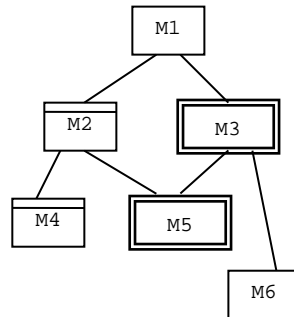
Los ciclos de vida clásicos definen un marco de trabajo controlado, de alta visibilidad sobre los procesos de desarrollo y un *estilo* de elaboración, en las diferentes fases, muy estable y repetitivo. Se denominan clásicos porque son los modelos de trabajo que se han venido empleando desde antes del origen de la Ingeniería de Software y porque están muy próximos a la manera *trivial* de elaborar cualquier producto. El ciclo de vida en cascada y en V es adecuado para la mayoría de los desarrollos cuya elaboración es suficientemente conocida y resulta especialmente cómodo para aquellos equipos de desarrollo u organizaciones que, por tener experiencia en la elaboración de un tipo de proyectos concreto, poseen procedimientos, plantillas o patrones de trabajo ajustados al desarrollo de dicho producto y que han demostrado sus ventajas en el éxito de anteriores proyectos de ese tipo. Sin embargo, estos modelos –ciclos de vida– se caracterizan, también, por su rigidez; ya que limitan el margen de maniobra ante circunstancias tales como los replanteamientos o la adecuación de las formas de trabajo descritas anteriormente. Las características de controlabilidad y visibilidad del desarrollo, de estos tipos de ciclo de vida, los hacen adecuados, también, para proyectos de gran envergadura en los que sea necesario tener un control y hacer un seguimiento cercano de sus partes. En las condiciones que se muestran en el enunciado, no parece aconsejable utilizar este modelo de trabajo.

El uso de modelos reducidos, maquetas y prototipos; su uso en simulación, análisis y pruebas para obtener experiencia o conclusiones que posibiliten paliar las incertidumbres que puedan existir en algún paso de la fabricación; ha sido una de las primeras y más notables contribuciones de la ingeniería al desarrollo de software. Actualmente existe un dinamismo frenético, no sólo en las herramientas, plataformas y tecnologías de desarrollo, sino en los entornos y tecnologías de ejecución y uso de los productos y servicios informáticos en sí. Por otro lado, el mercado demanda la necesidad de disminuir, drásticamente, los tiempos de desarrollo. El imperativo de la agilidad y la adaptabilidad, es la razón principal de que los ciclos de vida evolutivos y basados en prototipos, sean los más usados en los desarrollos actuales. Las ventajas del tratamiento eficaz de la incertidumbre y de la provisión de agilidad a los desarrollos, se contraponen a la disminución del control y de la visibilidad del proceso de fabricación, en relación con los ciclos de vida clásicos. En el escenario planteado en el enunciado, éste tipo de ciclo de vida parece el más adecuado.

El ciclo de vida en espiral se sitúa en un plano totalmente distinto respecto a los modelos anteriores; debido al hecho de que incorpora modelos de gestión al propio desarrollo. En este ciclo de vida, el modelo del

proceso de fabricación –que puede ser cualquiera de los ciclos de vida vistos anteriormente, en cada vuelta— está controlado a través de una gestión del riesgo, el cual acompaña todo el desarrollo del producto. Desde el punto de vista del enunciado, este modelo de trabajo no aporta nada nuevo respecto a lo argumentado anteriormente. La gestión del riesgo aumenta la controlabilidad entre etapas –sobre todo porque el análisis de riesgos genera una planificación que se coordina e integra en el Plan General del proyecto— aunque el incremento en la seguridad y estabilidad del desarrollo disminuya, quizás, su agilidad.

2. Dado el siguiente Diagrama de Abstracciones:

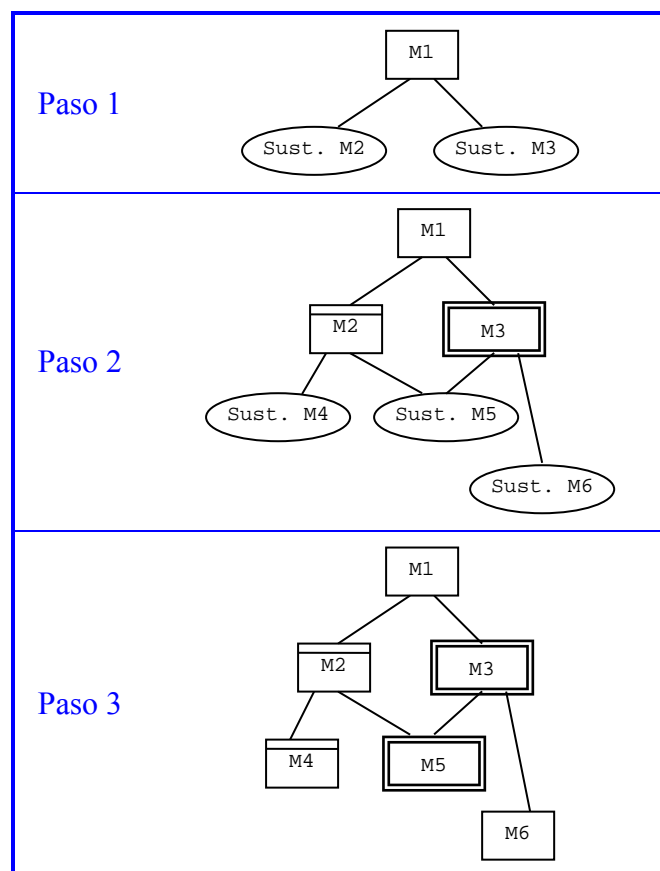


Establezca una comparativa entre las estrategias de integración ascendente y descendente, indicando el orden en el que se construirían los módulos del diagrama para cada una de ellas, así como sus ventajas e inconvenientes.

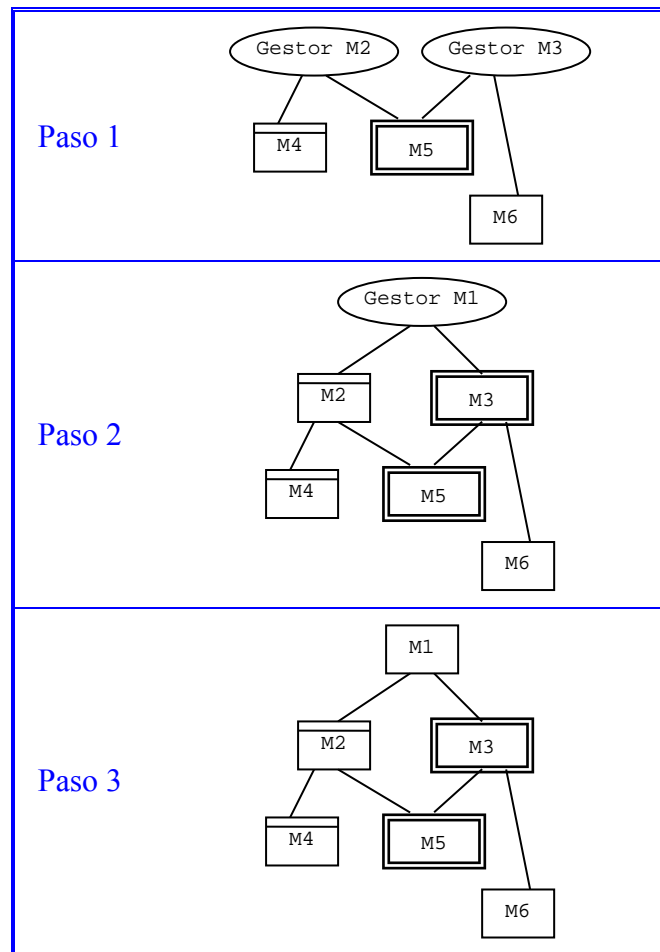
Solución

Examinemos el orden de realización de los módulos para cada estrategia de integración:

Integración Descendente



Integración Ascendente



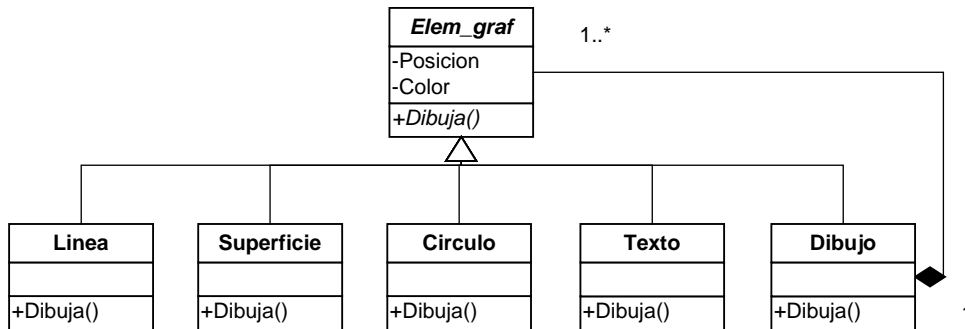
La siguiente tabla resume las ventajas e inconvenientes derivados del uso de las integraciones descendente y ascendente en el ejemplo que nos ocupa:

	I. Descendente	I. Ascendente
¿Facilita una visión general de la aplicación desde el principio?	Sí	No
Número de elementos de código desechables a construir	5 sustitutos o stubs	3 gestores o drivers
¿Facilita el ensayo de situaciones especiales para los módulos?	No	Sí
¿Facilita el trabajo en paralelo?*	Sí	Sí

(*): Aunque generalmente la integración ascendente propicia en mayor grado el trabajo en paralelo que la integración descendente, en este ejemplo, las dos estrategias de integración facilitan el trabajo en paralelo en el mismo grado.

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

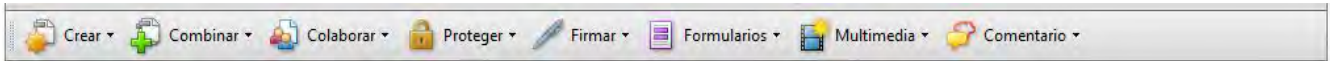
3. El diagrama de objetos de la figura es una solución que resuelve no sólo un problema específico de diseño, sino una buena cantidad de otros problemas similares. A eso se le denomina **patrón de diseño**.



El patrón anterior se denomina genéricamente **Contenedor** y corresponde, en este ejemplo, a la clase **Dibujo** de la figura.

- a. Supóngase que la clase **Decorador** es una especialización de **Dibujo** y la instancia del objeto `Mi_separador` es del tipo **Decorador**. La aplicación utilizaría el método `Mi_separador.Dibuja()`. **Explique qué ventajas tiene el uso de este patrón y de qué manera funciona como para justificar su uso extenso.**

Supóngase un entorno gráfico de ventanas como Windows donde los elementos que se manejan (**Objeto_entorno**) son una serie de elementos simples: Borde, Icono, Fondo, Etiqueta o Decorador; y otros **Elem_compuesto** son agrupaciones de los simples y de sí mismos, 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 desplegables, por Separadores y un Asa (estos últimos, los consideramos especializaciones de Decorador). Por ejemplo:



- b. Haciendo que el patrón de diseño '**Contenedor**' sea la clase **Elem_compuesto**, construya el diagrama de objetos correspondiente al entorno y la clase **Barra_herr**. Si se añade un botón de '**Zoom**' (**Boton_Zoom**) ¿Cómo cambia el diagrama?

Solución

El hecho de *conectar* la invocación de un método con el código que le corresponde, se llama **acoplamiento** o **enlace**. Cuando el compilador conoce exactamente qué código le corresponde a la invocación de un método, la asignación se denomina *enlazado temprano* (*early binding*) y se produce durante la compilación, antes de la ejecución. Sin embargo, si durante la compilación no se sabe qué código le corresponde exactamente en la invocación de un método, la asignación se resuelve durante la ejecución del programa, cuando las referencias toman un valor concreto. A esto se le llama *acoplamiento tardío* (*late binding*) o **enlazado dinámico**.

- a. El patrón Contenedor, **Dibujo**, sobrecarga los métodos de **Elem_graf**. Si se crea una instancia de tipo **Dibujo**, por ejemplo así:

```
Elem_graf Mi_Dibujo = new Dibujo()
```

se crea un objeto de tipo **Dibujo** e, inmediatamente, la referencia obtenida se asigna a un objeto, **Mi_Dibujo**, de tipo **Elem_graf**. El compilador no daría error puesto que, por la herencia, un **Dibujo** es, también, un **Elem_graf**. A esto se le llama **generalización** y es una transformación del tipo hacia arriba (*upcasting*) que se hace durante la compilación (enlazado temprano).

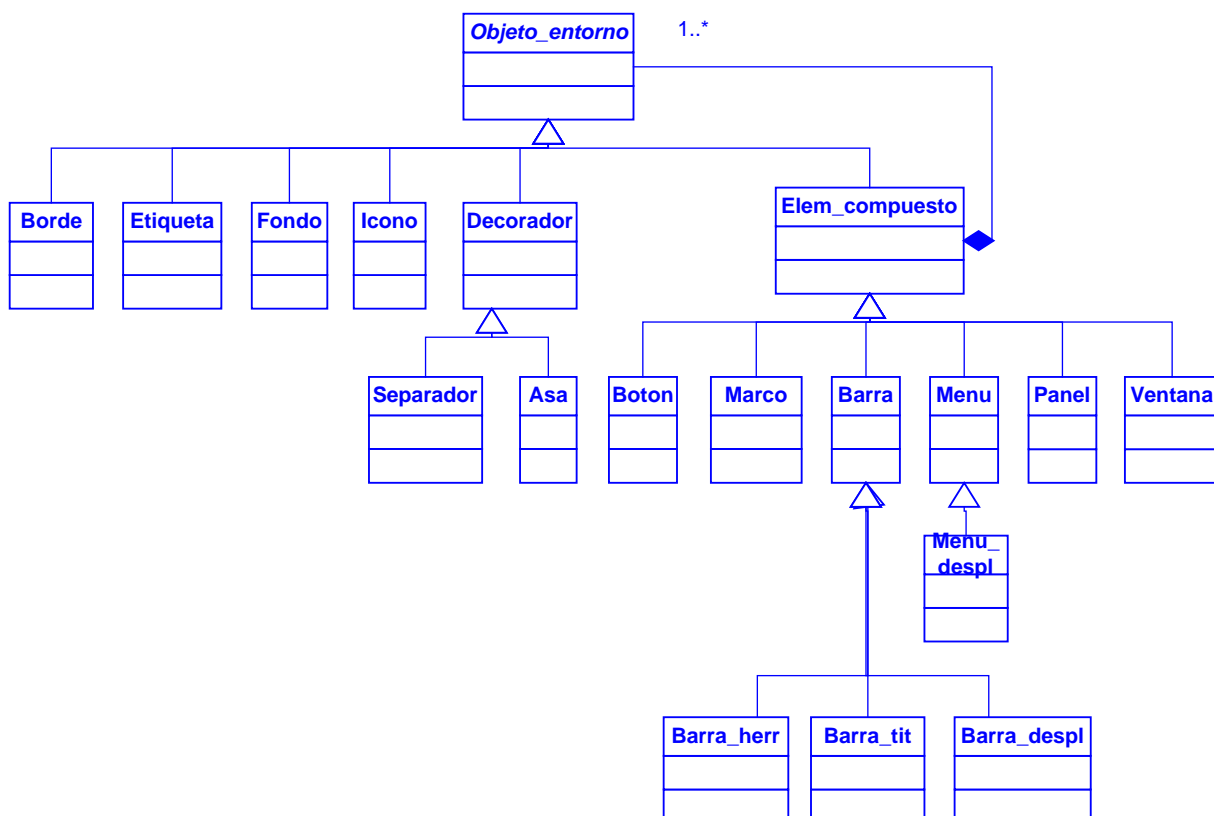
En el caso del patrón de composición, **Dibujo**, no sólo es un **Elem_graf** sino que, por la composición, puede estar compuesto por él mismo o cualquiera de sus hermanos y sus descendientes.

En el ejemplo, **Mi_Separador** es de tipo **Decorador** y, por la herencia, también es un **Dibujo** y un **Elem_graf**. Mediante la generalización, se le asignará —en tiempo de compilación— el tipo conocido del antecesor más cercano: **Dibujo** o **Elem_graf**, según haya sido la sentencia de creación.

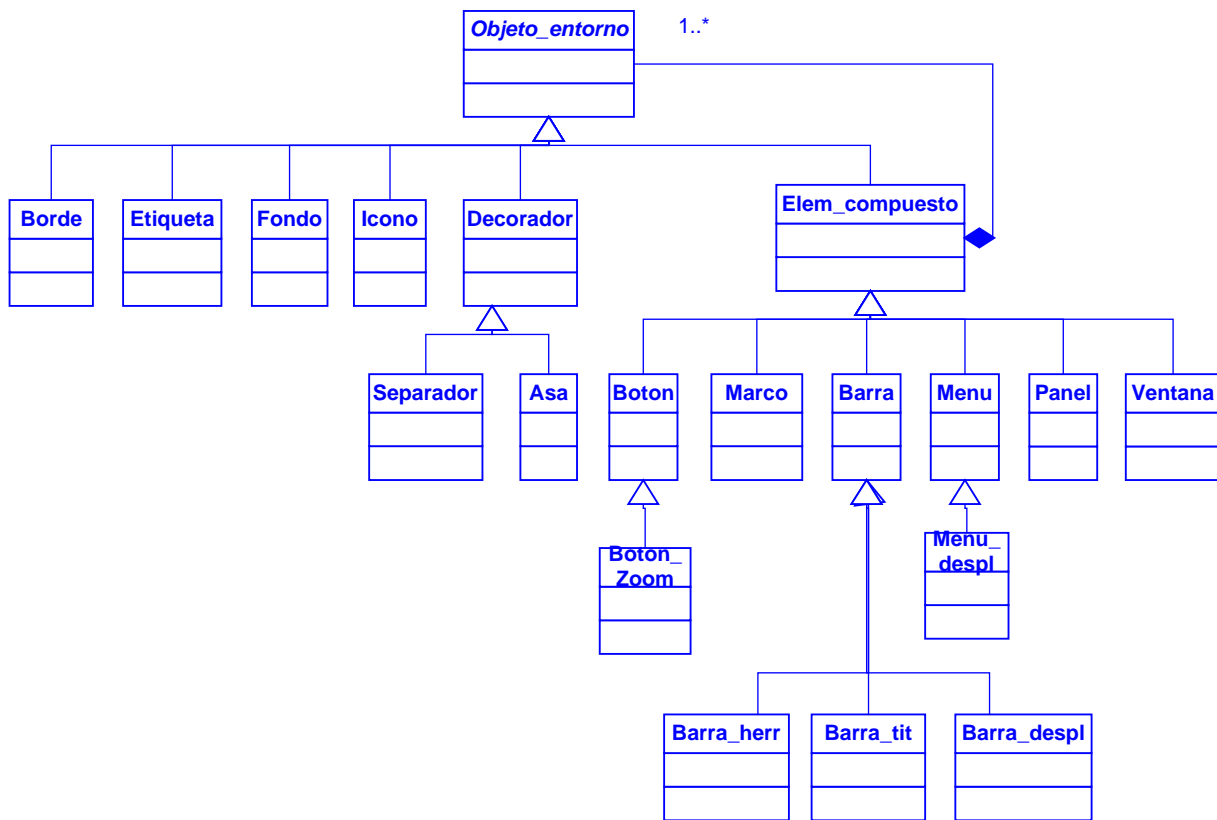
Al invocar, **durante la ejecución**, al método **Mi_Separador.Dibuja()**, el **polimorfismo** —o enlazado dinámico— garantiza que el método que se ejecuta en la invocación es la especialización correcta de **Decorador** —**Decorador.Dibuja()**— puesto que ese es el tipo de **Mi_Separador**.

Se pone de manifiesto de qué manera, la herencia y el polimorfismo, permiten diseñar con ocultación, ahorrar en el desarrollo de código —el cual resulta compacto, sencillo y fácilmente entendible— y lleva a diseños flexibles en los que es fácil añadir o eliminar elementos. El uso del patrón de composición tiene el valor añadido de que permite crear estructuras de cualquier profundidad, las cuales pueden ser perfectamente ignoradas para el código que las utilice —puesto que es suficiente con conocer la clase raíz, denominada *interfaz*— y, además, dicho código no se ve afectado porque se añadan elementos en la familia.

- b. El diagrama del entorno, con la clase **Barra_herr**, quedaría:



Si se añade la clase **Boton_Zoom**, quedaría:



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. Supóngase que una organización decide afrontar, por primera vez, un proyecto software en el que no tiene experiencia y que se sitúa en un ámbito desconocido para ella. Desde el punto de vista del ciclo de vida, indique qué alternativas tiene esta organización para culminar el proyecto con éxito y explique cómo pueden, dichas alternativas, mitigar los problemas potenciales con los que la organización se puede encontrar durante el desarrollo.

Solución

Los ciclos de vida clásicos definen un marco de trabajo controlado, de alta visibilidad sobre los procesos de desarrollo y un *estilo* de elaboración, en las diferentes fases, muy estable y repetitivo. Se denominan clásicos porque son los modelos de trabajo que se han venido empleando desde antes del origen de la Ingeniería de Software y porque están muy próximos a la manera *trivial* de elaborar cualquier producto. El ciclo de vida en cascada y en V es adecuado para la mayoría de los desarrollos cuya elaboración es suficientemente conocida y resulta especialmente cómodo para aquellos equipos de desarrollo u organizaciones que, por tener experiencia en la elaboración de un tipo de proyectos concreto, poseen procedimientos, plantillas o patrones de trabajo ajustados al desarrollo de dicho producto y que han demostrado sus ventajas en el éxito de anteriores proyectos de ese tipo. Sin embargo, estos modelos –ciclos de vida– se caracterizan, también, por su rigidez; ya que limitan el margen de maniobra ante circunstancias tales como los replanteamientos o la adecuación de las formas de trabajo descritas anteriormente. Las características de controlabilidad y visibilidad del desarrollo, de estos tipos de ciclo de vida, los hacen adecuados, también, para proyectos de gran envergadura en los que sea necesario tener un control y hacer un seguimiento cercano de sus partes. En las condiciones que se muestran en el enunciado, no parece aconsejable utilizar este modelo de trabajo.

El uso de modelos reducidos, maquetas y prototipos; su uso en simulación, análisis y pruebas para obtener experiencia o conclusiones que posibiliten paliar las incertidumbres que puedan existir en algún paso de la fabricación; ha sido una de las primeras y más notables contribuciones de la ingeniería al desarrollo de software. Actualmente existe un dinamismo frenético, no sólo en las herramientas, plataformas y tecnologías de desarrollo, sino en los entornos y tecnologías de ejecución y uso de los productos y servicios informáticos en sí. Por otro lado, el mercado demanda la necesidad de disminuir, drásticamente, los tiempos de desarrollo. El imperativo de la agilidad y la adaptabilidad, es la razón principal de que los ciclos de vida evolutivos y basados en prototipos, sean los más usados en los desarrollos actuales. Las ventajas del tratamiento eficaz de la incertidumbre y de la provisión de agilidad a los desarrollos, se contraponen a la disminución del control y de la visibilidad del proceso de fabricación, en relación con los ciclos de vida clásicos. En el escenario planteado en el enunciado, éste tipo de ciclo de vida parece el más adecuado.

El ciclo de vida en espiral se sitúa en un plano totalmente distinto respecto a los modelos anteriores; debido al hecho de que incorpora modelos de gestión al propio desarrollo. En este ciclo de vida, el modelo del proceso de fabricación –que puede ser cualquiera de los ciclos de vida vistos anteriormente, en cada

vuelta— está controlado a través de una gestión del riesgo, el cual acompaña todo el desarrollo del producto. Desde el punto de vista del enunciado, este modelo de trabajo no aporta nada nuevo respecto a lo argumentado anteriormente. La gestión del riesgo aumenta la controlabilidad entre etapas –sobre todo porque el análisis de riesgos genera una planificación que se coordina e integra en el Plan General del proyecto— aunque el incremento en la seguridad y estabilidad del desarrollo disminuya, quizás, su agilidad.

2. ¿Qué tres objetivos fundamentales o cualidades mínimas es deseable alcanzar al hacer la descomposición modular de un sistema? Explique cada uno de ellos y, en cada caso, cómo se pueden medir o qué factores intervienen.

Solución

Independencia funcional, comprensibilidad y adaptabilidad.

Síntesis y resumen esquemático del epígrafe 4.1. Páginas 148 a 160 del libro de texto.

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

3. Se desea construir un sistema que reciba como entrada una lista ordenada de líneas, cada una de las cuales está formada por una lista ordenada de palabras, que a su vez será una lista ordenada de caracteres. Sobre cada línea se pueden realizar rotaciones, que consisten en eliminar la primera palabra y concatenarla al final de la línea. El sistema devolverá como resultado una lista con las posibles rotaciones de todas las líneas ordenadas alfabéticamente (incluyendo las rotaciones nulas).

Por ejemplo,

<u>voy a cenar</u> <u>al restaurante</u>	→	<u>a cenar voy</u> <u>al restaurante</u> <u>cenar voy a</u> <u>restaurante al</u> <u>voy a cenar</u>
---	---	--

- a. Diseñe el sistema utilizando herencia, polimorfismo y Diagramas de Objetos.
- b. ¿Cómo cambiaría su diseño si el sistema recibe una línea –lista ordenada de palabras—, realiza rotaciones sobre cada palabra –va rotando los caracteres circularmente— y devuelve como resultado una lista con las posibles rotaciones de todas las palabras ordenadas alfabéticamente (incluyendo las rotaciones nulas)?

Por ejemplo,

<u>voy a cenar</u>	→	<u>a</u> <u>arcen</u> <u>cenar</u> <u>enarc</u> <u>narce</u> <u>oyv</u> <u>rcena</u> <u>voy</u> <u>yvo</u>
--------------------	---	--

Solución

En una primera aproximación, aplicaremos el método de Abbott para determinar las clases u objetos que componen el diseño. Para ello, marcaremos en rojo los sustantivos (candidatos a ser clases o atributos) y en azul los verbos (candidatos a ser métodos).

Se desea construir un sistema que reciba como entrada una *lista ordenada de líneas*, cada una de las cuales está formada por una *lista ordenada de palabras*, que a su vez será una *lista ordenada de caracteres*. Sobre cada línea se pueden *realizar rotaciones*, que consisten en *eliminar la primera palabra* y *concatenarla al final de la línea*. El sistema devolverá como resultado una *lista con las posibles rotaciones de todas las líneas ordenadas alfabéticamente* (incluyendo las rotaciones nulas).

A partir de este marcado elaboramos una doble lista con los elementos correspondientes a clases y a métodos.

CLASES O ATRIBUTOS	MÉTODOS
<ul style="list-style-type: none">• lista ordenada de líneas• lista ordenada de palabras• lista ordenada de caracteres• lista con las posibles rotaciones de	<ul style="list-style-type: none">• realizar rotaciones• eliminar la primera palabra• concatenar la primera palabra al final de la línea

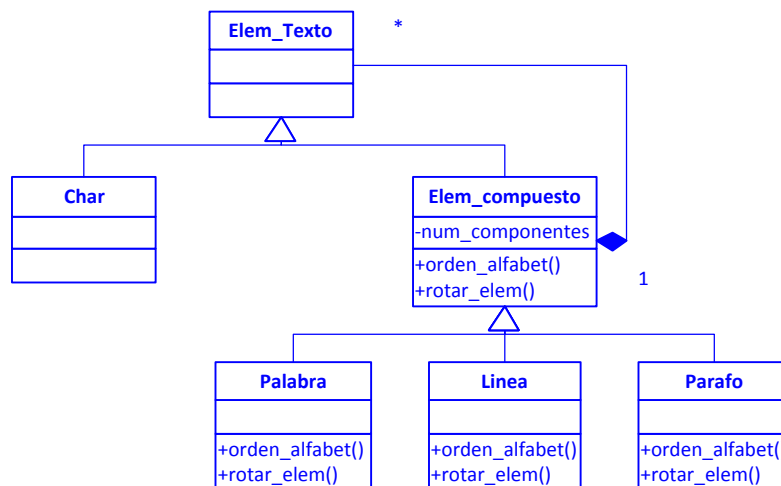
todas las líneas ordenadas
alfabéticamente



Cuando piense en cómo asignar una clase a una estructura de datos y en cómo distribuir la herencia o el polimorfismo, considere que una clase **es una definición del tipo de un dato**. Decir ‘Mirlo’ hereda de ‘Ave’ significa **que son del mismo tipo** y la relación se enuncia como: un ‘Mirlo’ ‘es un’ ‘Ave’. La especialización —y algunas formas de polimorfismo— también se deben a la herencia. Un ‘Avestruz’ ‘es como un’ ‘Mirlo’... Es pues un ‘Ave’ —tiene el mismo tipo— pero ha anulado el método de volar y tiene sobrecargadas otras funciones de ‘Ave’.

Del análisis del cuadro anterior, obtenido con el método de Abbott, parece claro que se van a manejar listas ordenadas cuyos componentes son elementos de texto. El objetivo de la aplicación es realizar determinadas operaciones con lo que hemos denominado ‘listas ordenadas’. Ahora bien, cada elemento que se maneja en una lista es, a su vez, otra lista; a no ser que el elemento sea un carácter. Además, para ahorrar código y simplificar el diseño, sería conveniente que el código escrito para una operación en una ‘lista ordenada’ —por ejemplo, ‘ORDENAR ALFABÉTICAMENTE LAS LÍNEAS DE ENTRADA’— sirva igualmente para otras listas. Al fin y al cabo, ¡todos los elementos de una lista son elementos de texto! Éste es el elemento crucial: encontrar el *tipo común*.

El diseño podría ser así:



ESTE EJERCICIO ES DE TIPO MIXTO.

ES IRRELEVANTE SI CONTESTA EN LA CASILLA DE TEST O NO (lo que ponga, NO CUENTA para la calificación). 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. Defina y distinga la 'validación' y la 'verificación'. ¿En qué fase del ciclo de vida en cascada se realiza cada una?

Solución

Libro de texto, epígrafe 1.4.1.2, páginas 15 y 16.

Validación: la comprobación de que un elemento (o todo el sistema) satisface las necesidades del usuario identificadas durante el análisis.

Verificación: la comprobación de que una parte (módulo o unidad) del sistema cumple con sus especificaciones particulares (de funcionamiento).

Evidentemente, estas comprobaciones se deberían hacer siempre; independientemente del ciclo de vida que se esté utilizando. La validación se hará después de la integración y pruebas de una parte funcionalmente significativa del sistema o del sistema completo, antes de su entrega y aceptación por parte del cliente. Por el contrario, en la verificación se comprueba que una unidad funcional se ejecuta correcta y coherentemente con las especificaciones de funcionamiento que se han establecido en el diseño para esa parte de la aplicación. Por tanto, la verificación se confunde con las '*pruebas de unidades*' y se hace tras la codificación o, si se está comprobando un módulo, tras la integración de sus componentes.

2. Defina los siguientes tipos de pruebas de software: *caja negra*, *caja transparente*, *alfa* y *beta*. ¿Las pruebas *alfa* y *beta* son de caja negra o transparente? Y las pruebas de unidades ¿de qué tipo son y cuándo se hacen?

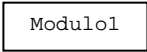

Solución

Epígrafes 5.7.2, 5.7.3 y 5.9.2 del libro de texto.

Las pruebas *alfa* y *beta* son pruebas del sistema completo, integrado. Pueden formar parte de las comprobaciones establecidas para la validación. Son de caja negra. Para las pruebas de unidades se puede seguir cualquier técnica de las reflejadas en los epígrafes de 5.7: de caja negra o de caja transparente. Estas pruebas constituyen la *verificación* y se hacen tras la codificación o, si se está comprobando un módulo, tras la integración de sus componentes.

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

3. Dentro de un sistema informático se emplea el módulo `Modulo1`. Para su desarrollo se plantean dos diseños alternativos:

Diseño 1	Diseño 2
	
<div>Código aproximado en Modula-2</div> <pre>1MODULE Modulo1; 2FROM InOut 3 IMPORT WriteCard, WriteString; 4... 5VAR 6 fecha1, fecha2: 7 ARRAY [1..6] OF TipoDigito; 8 ... 9BEGIN 10 ... 11 (* fecha1 := 30-04-1974 *) 12 fecha1[1] := 3; 13 fecha1[2] := 0; 14 fecha1[3] := 0; 15 fecha1[4] := 4; 16 fecha1[5] := 7; 17 fecha1[6] := 4; 18 (* Imprimir anno *) 19 WriteCard(fecha1[5], 1); 20 WriteCard(fecha1[6], 1); 21 ... 22 (* fecha2 := 10-01-1998 *) 23 fecha2[1] := 1; 24 fecha2[2] := 0; 25 fecha2[3] := 0; 26 fecha2[4] := 1; 27 fecha2[5] := 9; 28 fecha2[6] := 8; 29 ... 30END Modulo1.</pre>	<pre>1MODULE Modulo1; 2IMPORT Fecha; 3... 4VAR 5 fecha1, fecha2: Fecha.Tipo; 6 ... 7BEGIN 8 ... 9 Fecha.Crear(fecha1, 30, 4, 1974); 10 Fecha.ImprimirAnno(fecha1); 11 ... 12 Fecha.Crear(fecha2, 10, 1, 1998); 13 ... 14END Modulo1. 1DEFINITION MODULE Fecha; 2TYPE Tipo; 3PROCEDURE Crear(VAR fecha: Tipo); 4PROCEDURE ImprimirAnno(fecha: Tipo); 5... 6END Fecha.</pre>

Compare los dos diseños analizando cómo aplican los conceptos de Abstracción, Modularidad y Ocultación.

Inicialmente, cuando se planteó el sistema informático, se consideró que para el almacenamiento de los años bastaba con dos dígitos. Sin embargo, con la llegada el nuevo milenio se descubrió que eran necesarios cuatro dígitos. **Razone cómo afectaría este cambio a cada uno de los diseños.**

Solución

1. Abstracción

- El primer diseño no utiliza abstracciones de ningún tipo.
- El diseño 2 utiliza el Tipo Abstracto de Datos¹ `Fecha`.

Como resultado, se puede observar que el código asociado al diseño 1 será muy redundante (líneas 12-17 ≈ líneas 23-28). La redundancia induce a errores e inconsistencias.

¹Concretamente y según la nomenclatura estudiada en la asignatura Programación 1, el *Tipo Opaco de Datos* `Fecha`.

2. Modularidad

El diseño 1 es monolítico, mientras que el diseño 2 es modular. Por ello, el diseño 2 dispone de las siguientes ventajas sobre el diseño 1:

- Permite dividir la implementación entre varias personas (un programador puede codificar `Modulo1` y otro, `Fecha`)
- La implementación asociada al diseño 2 es clara y concisa.
- Los costes asociados al desarrollo, la depuración, la documentación y el mantenimiento del diseño 2 son menores que los del diseño 1.
- El diseño 2 permite reutilizar el concepto de fecha en otros proyectos.

3. Ocultación

En el diseño 1 las “interioridades de las fechas están al descubierto”, es decir, no existe ocultación del concepto de fecha. Este hecho, conlleva dos grandes problemas:

- Si se produce un error en el uso de una fecha, su detección será ardua, ya que a este concepto se accede directamente desde muchos puntos del programa.
- La modificación de la representación del concepto fecha se propagará a muchas partes del código del `Modulo1`. Si tal y como se propone en el enunciado, se decide extender la representación de los años de 2 dígitos a 4, será necesario modificar las líneas 7, 12-28.

En el diseño 2, el concepto de fecha se encapsula y oculta a través de un Tipo Abstracto de Datos. Las consecuencias benignas de esta estrategia son:

- Si se produce un error asociado a una fecha, la búsqueda se limitará al módulo de implementación de `Fecha`.
- Gracias a la ocultación del concepto fecha, la ampliación del número de dígitos de los años implica cambios exclusivamente en el código del módulo de implementación de `Fecha`.

ESTE EJERCICIO ES DE TIPO MIXTO.

ES IRRELEVANTE SI CONTESTA EN LA CASILLA DE TEST O NO (lo que ponga, NO CUENTA para la calificación). 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. Suponga que se quiere modelar un sistema en el que lo más importante es representar las tareas y transformaciones que se hacen con la información y los demás aspectos son prácticamente irrelevantes. ¿Sería imprescindible el uso de Diagramas Entidad-Relación? ¿Qué aspecto del modelo prioriza esta notación?

Solución

Epígrafes 2.3.2, 2.3.3 y 2.3.6 del libro. Páginas 52 a 60 y 65 a 68.

Los tres tipos diagramas para representar de modelos del comportamiento del sistema, que se ven en la asignatura, son complementarios y cuasi-ortogonales. Cada uno de ellos hace hincapié en un grupo de aspectos del modelo, diferenciado de los otros. De esta forma, para que un sistema esté completa o suficientemente descrito, necesitaría de todas las vistas. Ahora bien, el objetivo principal de un modelo es que facilite la comprensión del comportamiento del sistema que se va a construir; y lo haga hasta tal punto que se pueda diseñar el funcionamiento con una incertidumbre mínima y, después, la implementación del diseño lleve al comportamiento deseado. Por tanto, el grado de desarrollo del modelo, dependerá de la complejidad del sistema y de los aspectos que aún no se comprenden o no están definidos con claridad. En este caso, para representar las tareas y transformaciones que se hacen con la información, lo más apropiado es utilizar Diagramas de Flujo de Datos. Si, con esta representación, se alcanza la comprensión del comportamiento que se espera del producto y los demás aspectos son, realmente, irrelevantes, se podría prescindir de DTE y DER.

Los Diagramas Entidad-Relación son modelos de datos que se centran en los aspectos relacionados con los '*objetos*', '*artefactos*' o elementos de información –las entidades— y en las vinculaciones relevantes que existen entre ellos –las relaciones—; pero no en cómo se manejan ni cómo se controla el funcionamiento del sistema ni cuál es su evolución dentro de él.

2. ¿Qué tres objetivos fundamentales o cualidades mínimas es deseable alcanzar al hacer la descomposición modular de un sistema? Explique cada uno de ellos y, en cada caso, cómo se pueden medir o qué factores intervienen.

Solución

Introducción de la sección “4.1 Descomposición modular” y epígrafes 4.1.1 a 4.1.3. Páginas 149 a 160 del libro de la asignatura.

Realizar la descripción del funcionamiento del sistema descomponiéndolo en módulos, tiene como objetivo fundamental facilitar la implementación y que, tras la codificación, disminuyan los problemas de funcionamiento. El diseño *‘hace funcionar’* al producto porque define sus mecanismos de funcionamiento, que se codificarán a continuación. Pero las decisiones relativas a **qué** diseño se utiliza, qué descomposición modular se elige, están orientadas a facilitar el mantenimiento, inmediato o futuro. Que el producto sea más o menos mantenible está estrechamente ligado a los costes de producción y a la rentabilidad que se espera al comercializarlo.

Para incrementar la mantenibilidad del software, la experiencia acumulada en desarrollos anteriores aconseja que la descomposición modular elegida cumpla con unos niveles mínimos de:

1. Independencia funcional. (Qué es y cómo se mide. Síntesis de las páginas 150 a 157 del libro)
2. Comprensibilidad.
3. Adaptabilidad

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

3. Con el fin de promocionar el uso del transporte público y el ocio al aire libre, RENFE ha decidido encargar la construcción de un sistema informático que asesore a sus clientes acerca de *‘rutas verdes’* para hacer a pie a partir de sus estaciones de tren.

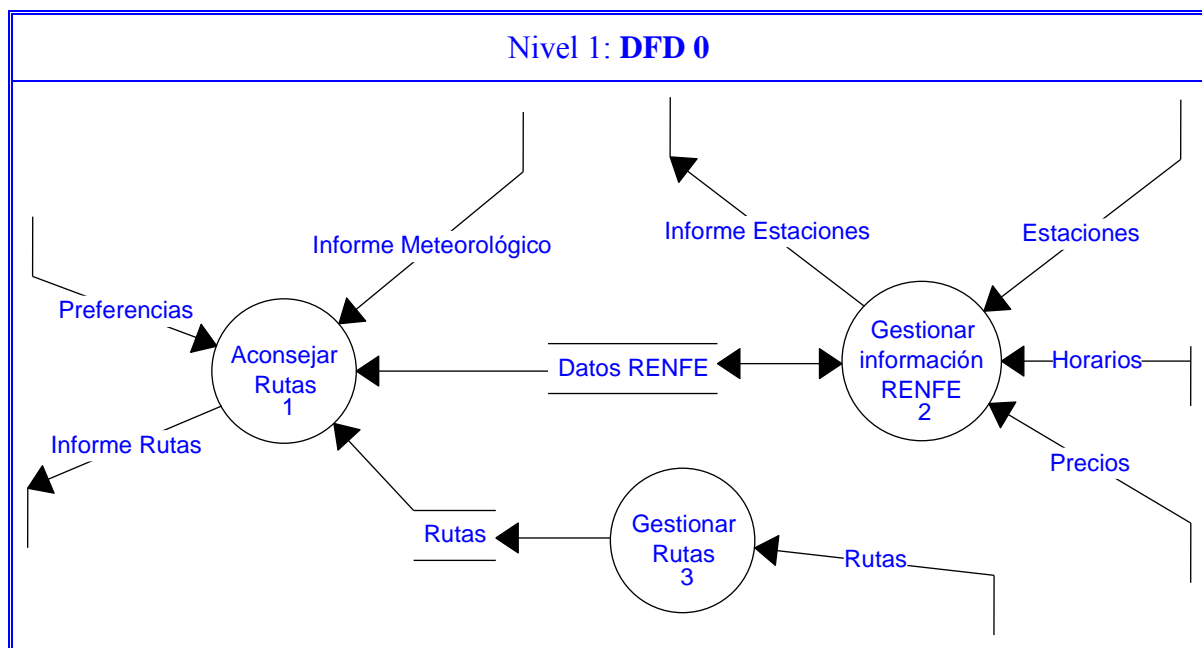
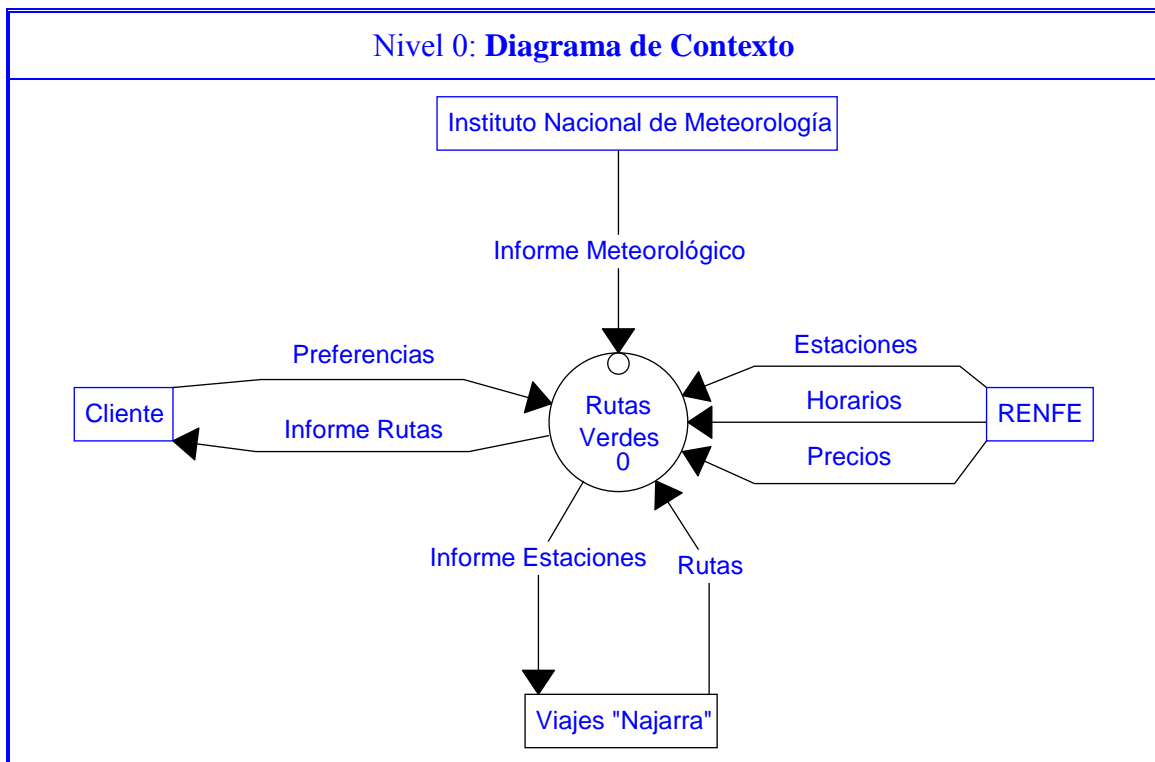
El sistema recibirá periódicamente la siguiente información:

- Un informe meteorológico del Instituto Nacional de Meteorología que contendrá las previsiones climáticas para los próximos días.
- Datos referentes a las estaciones de tren, horarios y precios de billetes. Esta información será suministrada por RENFE.
- Se ha encargado a la empresa “Viajes Najarra” la elaboración e introducción en el sistema de las rutas verdes. Para ello, la empresa podrá solicitar del sistema un informe de las estaciones de RENFE existentes.

Los clientes introducirán en el sistema sus preferencias. A partir de estas y los datos antes descritos, se construirá un informe con las rutas aconsejadas.

Analice el sistema mediante DFDs (Diagramas de Flujo de Datos), desarrollando exclusivamente los DFDs de nivel 0 (contexto) y nivel 1.

Solución



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)

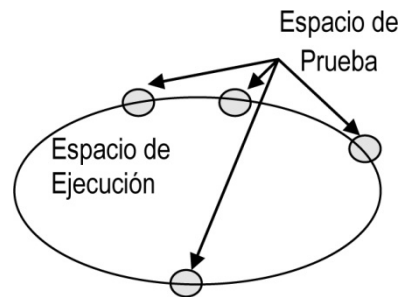
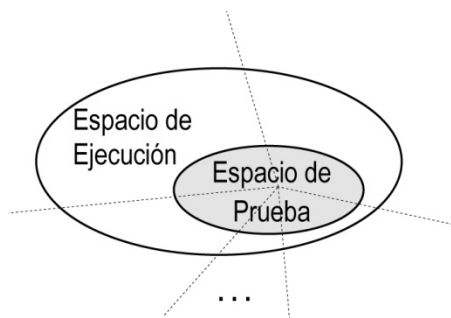
PRIMERA PARTE. PREGUNTAS TEÓRICAS (2'5 PUNTOS CADA UNA)

1. ¿Qué es el Análisis de Dominio y para qué se usa?

Solución

Epígrafes 2.1.2.5, en página 40, y 2.2.2 en la página 46.

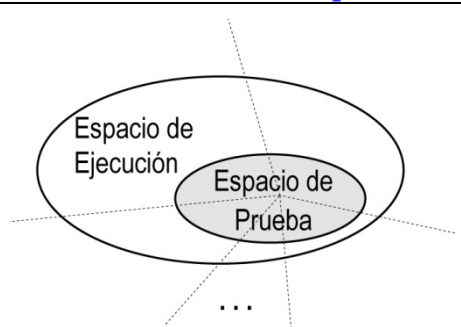
2. Las siguientes figuras representan dos métodos de prueba de caja negra: el análisis de valores límite y la partición en clases de equivalencia.
 - Estos métodos limitan el *espacio de prueba* a una parte del *espacio de ejecución*, ¿por qué?
 - Identifique qué método corresponde a cada figura y resuma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

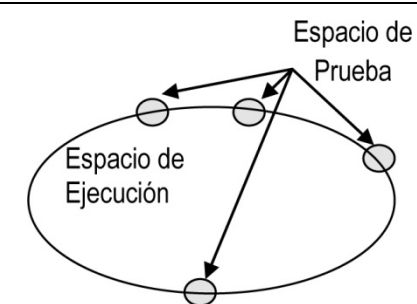
b)

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. Un sistema informático de punto de venta (**PDV**) se suele utilizar en muchas tiendas para registrar ventas y realizar pagos. Tiene componentes hardware, como un terminal y un lector para los identificadores de los artículos. El software interactúa con módulos de terceras partes, que dan servicios de captura de facturación y cálculo de impuestos, de control del inventario y de cálculo de comisiones y márgenes de beneficio para el vendedor en el sistema de contabilidad. Una operación típica de este PDV es “**Procesar Venta**”, cuyo *escenario principal de éxito* (o **flujo básico**) se puede expresar, en *formato breve*:

Procesar Venta: Un cliente llega a la caja de una tienda para comprar varios artículos. El cajero utiliza el sistema PDV para registrar los artículos. El sistema presenta una suma parcial y detalles de cada línea de venta. El cliente introduce los datos del pago y el sistema los valida y registra. El sistema actualiza el inventario. El cliente recibe un recibo del sistema y se va con sus artículos.

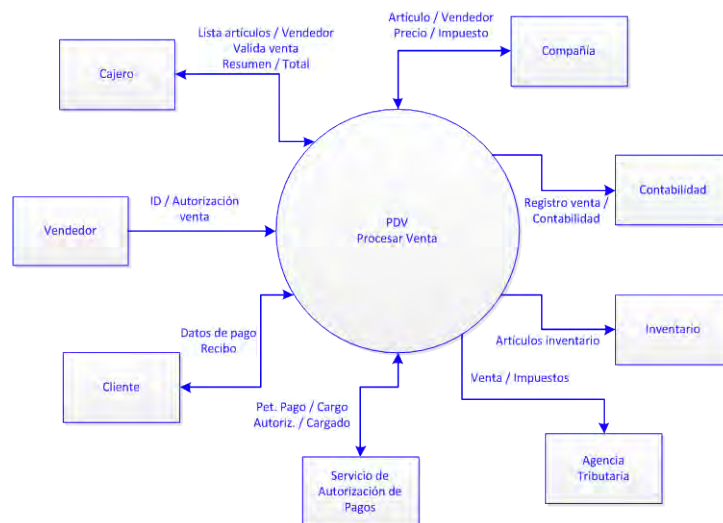
Reflexione sobre los intereses (necesidades) de cada uno de los actores involucrados en la operación:

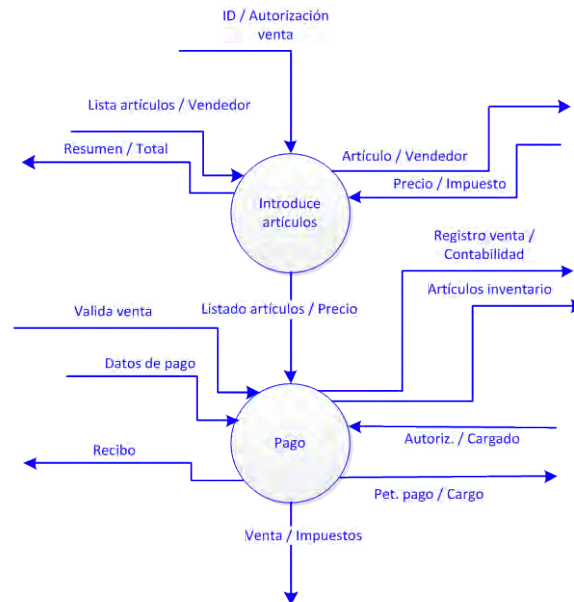
- **Compañía:** Quiere registrar las transacciones con precisión y satisfacer los intereses de los clientes. Quiere asegurar que se registran los pagos aceptados por el Servicio de Autorización de Pagos. Quiere actualización automática y rápida de la contabilidad y el inventario.
- **Cajero:** Quiere entradas precisas, rápidas y sin errores de pago, pues las pérdidas se deducen de su salario.
- **Cliente:** Quiere hacer el pago sin problemas y obtener el recibo correcto.
- **Vendedor:** Quiere actualizadas las comisiones de las ventas.
- **Agencia Tributaria:** Quiere recopilar los impuestos de cada venta.
- **Servicio de Autorización de Pagos:** Quiere recibir peticiones de autorización digital con el formato y protocolo correctos. Quiere registrar, de manera precisa, las cuentas por cobrar de la tienda.

- a. Construya un modelo de comportamiento para la operación **Procesar Venta**, descrita en su flujo básico, mediante DFDs (contexto y nivel 1). Explíquelo.

Solución

Considerando la mayoría de los intereses de los actores involucrados en la operación, los DFD de nivel 0 y 1 son:



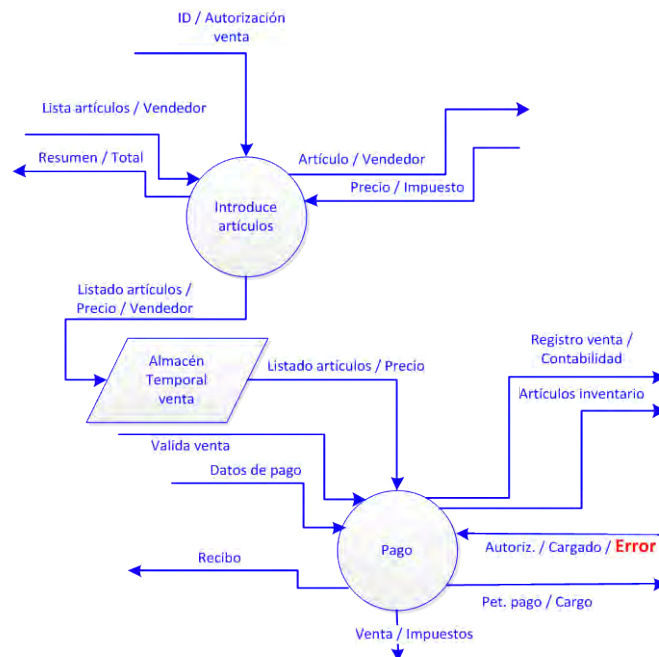


b. Qué solución aportaría al modelo anterior, si las necesidades de la compañía se modifican de la siguiente manera:

- **Compañía:** Quiere registrar las transacciones con precisión y satisfacer los intereses de los clientes. Quiere asegurar que se registran los pagos aceptados por el Servicio de Autorización de Pagos. Quiere cierta tolerancia a fallos que permita capturar las ventas, incluso si algún componente del servidor (ej. la validación remota del crédito) no están disponibles. Quiere actualización automática y rápida de la contabilidad y el inventario.

Vuelva a representar el DFD 0 anterior, pero con este nuevo requisito. Explíquelo.

Solución



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. ¿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.

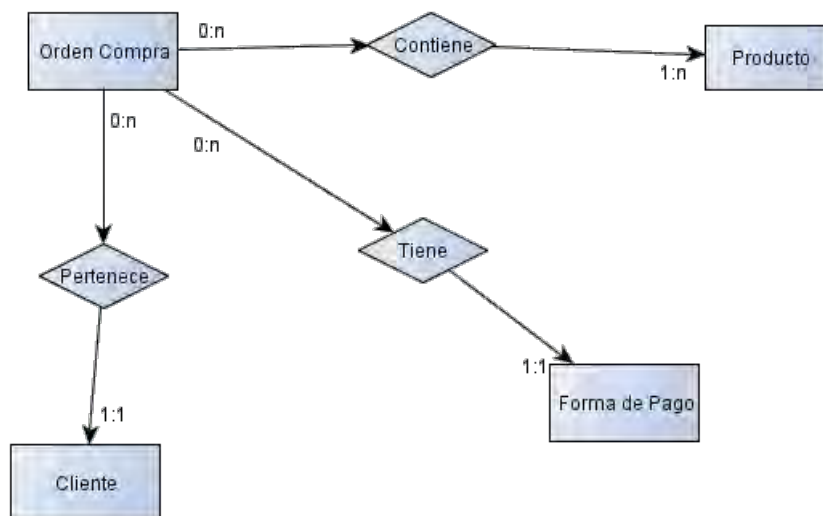
En ocasiones es necesaria para el mantenimiento de dichos productos software. (Pág. 26)

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 + Forma de Pago + {Producto}**
Identificador = {CaracterAlfanumérico}^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 = {CaracterAlfanumérico}¹⁰ /ristra de 10 caracteres /

Apellidos = {CaracterAlfanumérico}³⁰ /ristra de 30 caracteres/

IdDNI = {Dígito}⁸

Usuario = {CaracterAlfanumérico}¹⁰

Clave = {CaracterAlfanumérico}¹⁰

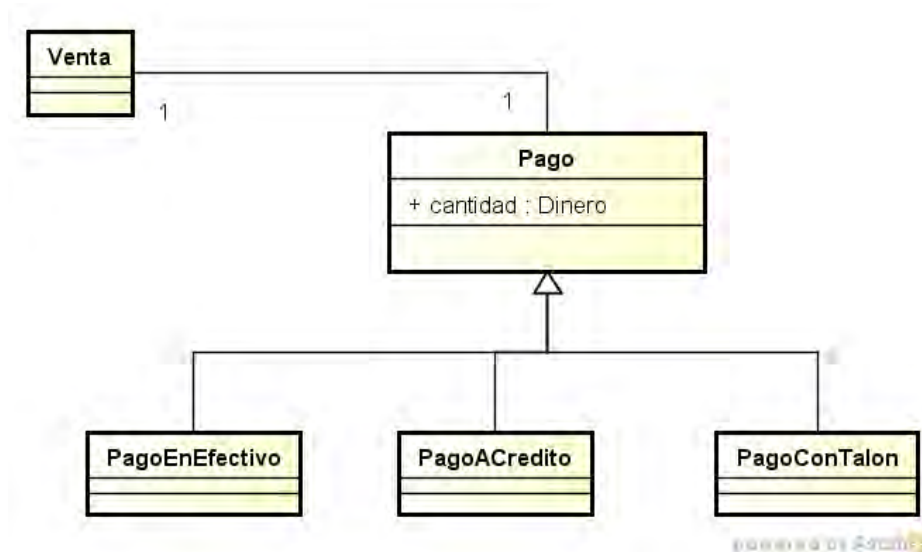
Dirección = {CaracterAlfanumérico}⁵⁰

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

3. Un sistema informático de punto de venta (**PDV**) se suele utilizar en muchas tiendas para registrar ventas y realizar pagos. Una operación típica de este PDV es “**Procesar Venta**” en la que, tras registrar los artículos y calcular el importe, el proceso de **Venta** invoca la ejecución del **Pago**. Ahora bien, el **Pago** puede ser **EnEfectivo**, **ACredito** o **ConCheque**. Al hacer el diseño, se ha creado una clase **Venta** asociada a la clase **Pago**.

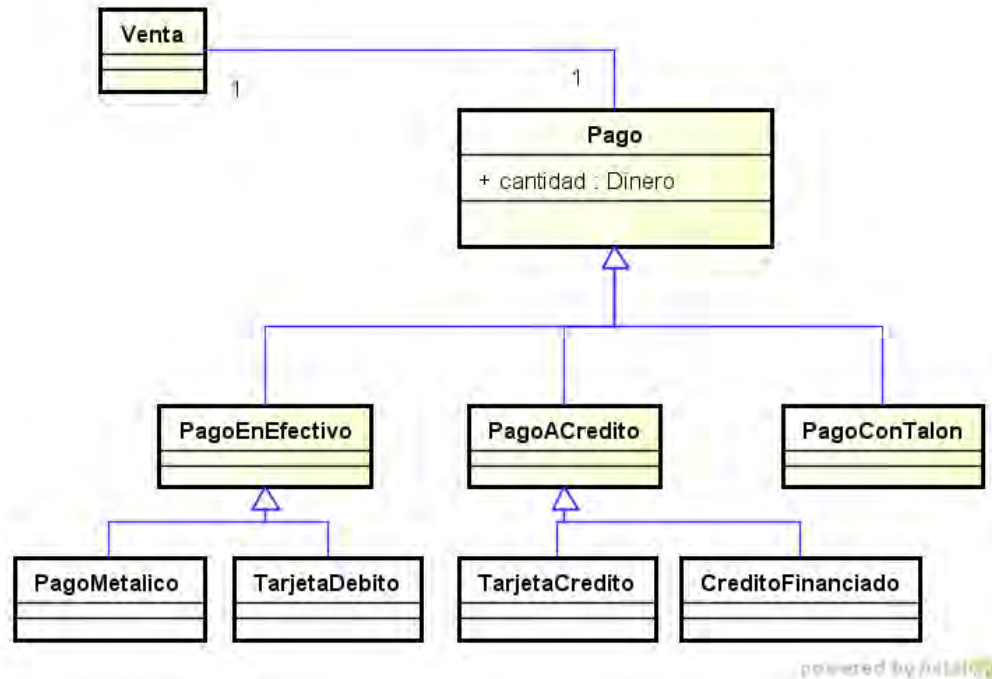
- a. Construya un modelo de objetos para las modalidades de pago.

Solución



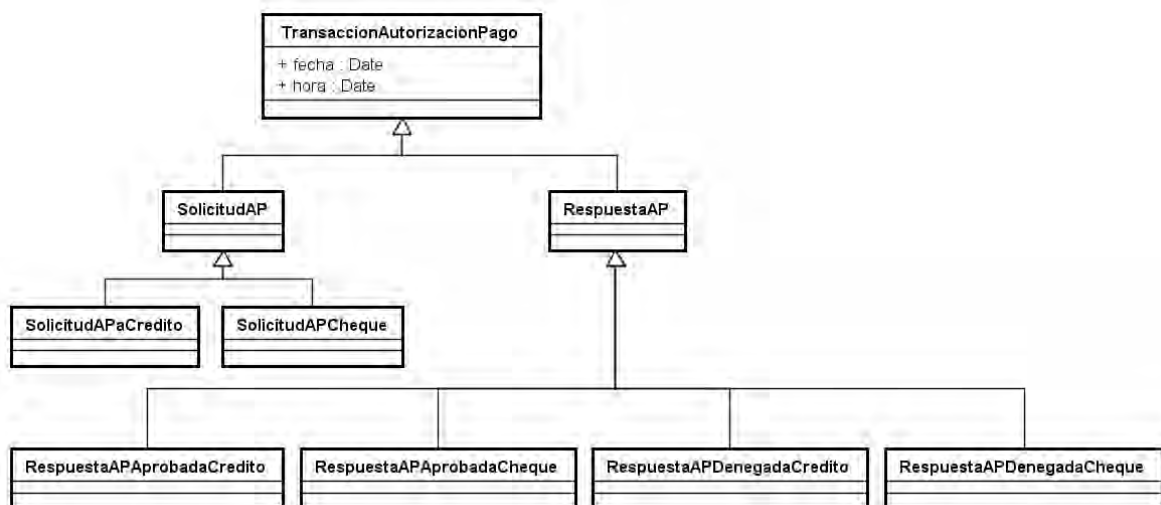
- b. Suponga que también se quiere considerar el pago en metálico, con tarjeta de débito, a crédito con tarjeta de crédito y crédito financiado. Construya el nuevo diagrama.

Solución



- c. En alguna de estas modalidades de pago, su ejecución implica Transacciones de Autorización del Pago. Una **TransaccionAutorizacionPago** es una **SolicitudAutorizacionPago** o bien una **RespuestaAutorizacionPago**. A su vez, la respuesta puede ser **RespuestaAprobadaPago** o **RespuestaDenegadaPago**. Es obvio que cada transacción se comporta de distinta forma con cada modalidad de pago. Para las modalidades de Pago ACredito y ConCheque, construya la jerarquía de clases partiendo de **TransaccionAutorizacionPago**; de manera que tenga la máxima simplicidad y flexibilidad. ¿Cuántas clases hay que añadir si incluimos la comprobación de saldo en una tarjeta de débito?

Solución



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.

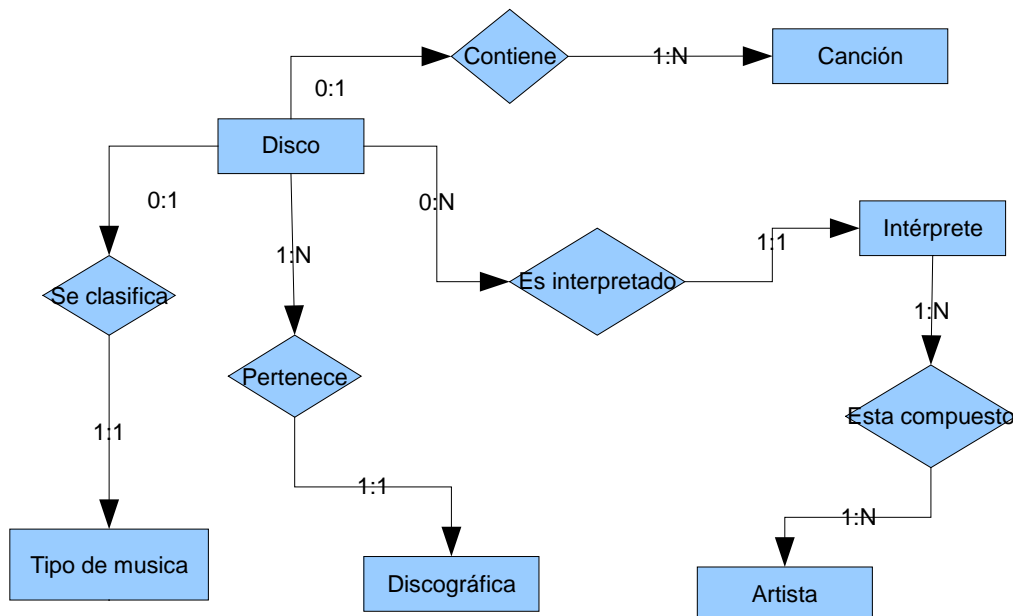
Conteste a las preguntas teóricas, en cualquier orden, en hojas diferentes a las que utilice para la contestación de la segunda parte. En cada parte, la cantidad MÁXIMA de papel (de examen, timbrado) que puede emplear ESTÁ LIMITADA al equivalente a DOS (2) HOJAS de tamaño A4 (210 x 297 mm)

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

- La gestión del catálogo de discos de una biblioteca se debe basar en la siguiente especificación:

Un disco tiene un título y un intérprete. El intérprete puede ser un artista sólo o un grupo (conjunto de artistas). El disco pertenece a una discográfica y contiene una serie de canciones. Así mismo, cada disco puede ser clasificado según un tipo de música.

Modele el enunciado anterior mediante un **Diagrama Entidad-Relación**. Describa los principales datos utilizando la notación del **Diccionario de Datos**.

Solución

Diccionario de datos:

DISCO = Nombre + INTERPRETE + TIPO-DE-MÚSICA + DISCOGRÁFICA + LISTA-DE-CANCIONES + año

INTERPRETE = ARTISTA

ARTISTA = {nombre}

DISCOGRÁFICA = nombre


TIPO-DE-MÚSICA = [jazz | clásica | pop | rock | heavy | dance]

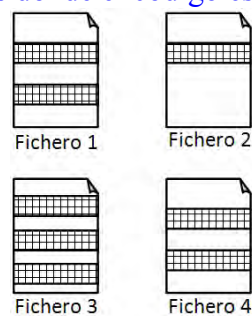
CANCIÓN = nombre + ARTISTA



LISTA-DE-CANCIONES = {CANCIÓN}

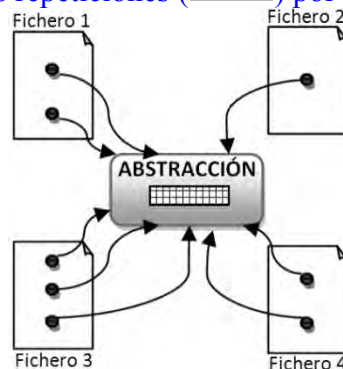
2. Una práctica muy extendida para reutilizar software es “cortar y pegar” fragmentos de código. Esto puede llevar a que un trozo de código se encuentre repetido muchas veces en un programa. ¿Cómo afecta esto al mantenimiento del programa? ¿Cómo podrían solventarse los efectos negativos del código duplicado?

Solución

La duplicidad de código perjudica la adaptabilidad de un programa. Por ejemplo, la siguiente figura representa un programa compuesto por 4 ficheros que contienen cierto código repetido (representado por ). Si durante el mantenimiento del programa se decide modificar dicho código repetido, habrá que realizar el cambio en todos los lugares donde se encuentra el código (en el ejemplo, habría que repetir el cambio en 8 puntos). Este trabajo no sólo es tedioso, si no que puede llevar a inconsistencias cuando los cambios no se aplican a todos los lugares donde el código está repetido.



Como indica la siguiente figura, estos problemas podrían evitarse encapsulando el código repetido en algún tipo de abstracción y sustituyendo las repeticiones () por invocaciones () a la nueva abstracción.



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

3. Se ha codificado en Modula-2 el siguiente subprograma que distingue, por el tamaño de sus lados si un triángulo es isósceles (dos lados iguales), equilátero (todos los lados iguales) o escaleno (ningún lado igual).

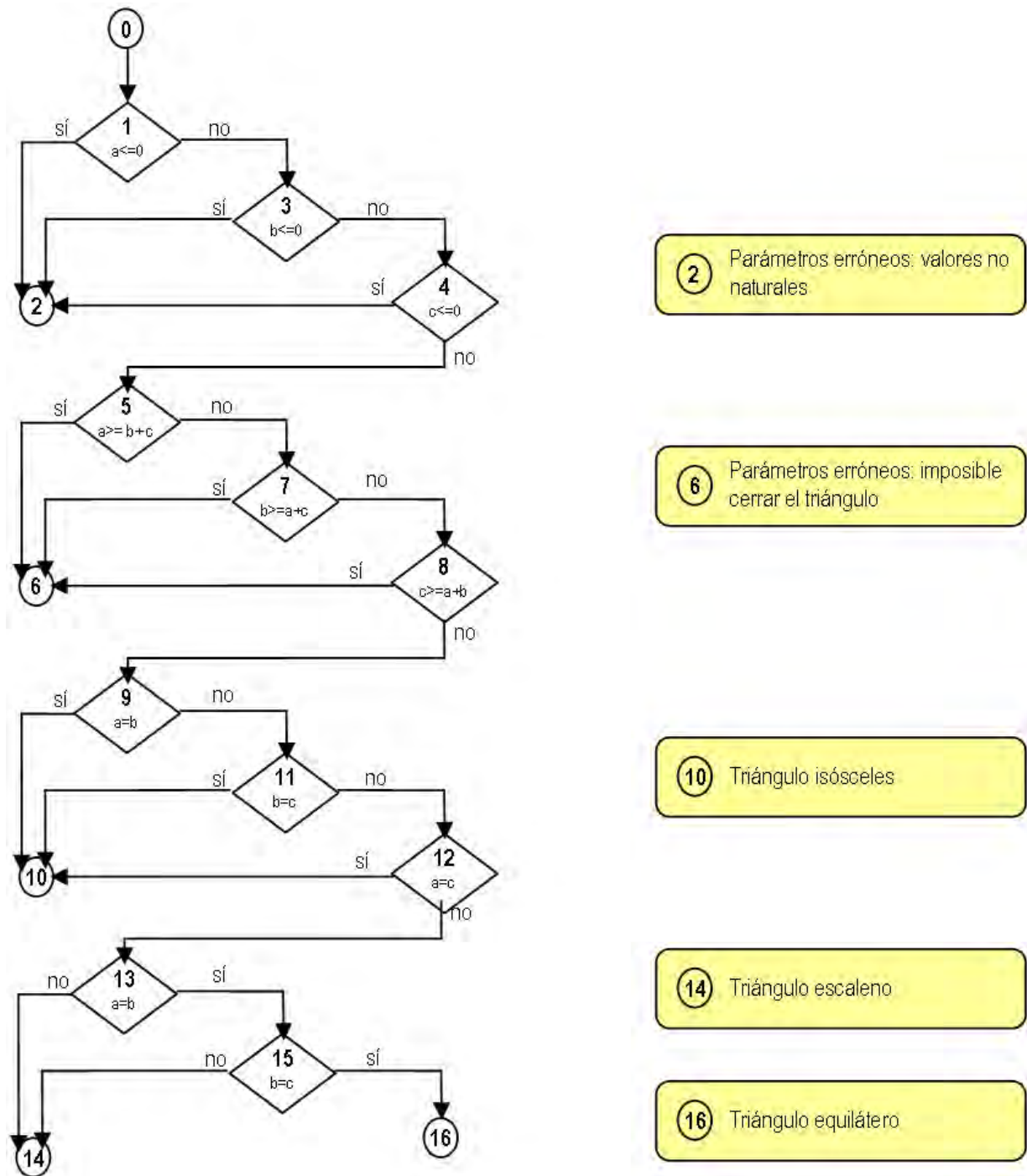
```
PROCEDURE TipoDeTriangulo(a, b, c: INTEGER);
BEGIN
  IF (a<=0) OR (b<=0) OR (c<=0) THEN
    WriteString("Error: alguno de los parámetros no es un número natural");
  ELSIF (a>=b+c) OR (b>=a+c) OR (c>=a+b) THEN
    WriteString("Error: no es posible cerrar el triángulo");
  ELSIF (a=b) OR (b=c) OR (a=c) THEN
    WriteString("El triángulo es isósceles");
  ELSIF (a=b) AND (b=c) THEN
    WriteString("El triángulo es equilátero");
  ELSE
    WriteString("El triángulo es escaleno");
  END;
END TipoDeTriangulo;
```

Verifique el subprograma con pruebas de caja transparente, realizando el cubrimiento lógico correspondiente.

Solución

Para la verificación del subprograma se elaborará un conjunto de casos de prueba que consigan que se transite por todos los posibles caminos de ejecución y que pongan en juego todos los elementos del código.

En primer lugar, el código del subprograma se transformará en el siguiente diagrama de flujo, donde cada rombo representa un predicado lógico simple:



A continuación, se calculará el nº de caminos básicos para recorrer todas las líneas de flujo del diagrama al menos una vez:

Nº de predicados = 11

Nº máximo de caminos = Nº de predicados + 1 = 12

A continuación se determinarán los caminos y los casos de prueba que forzarán su recorrido.

Como indica la siguiente figura, es imposible escribir un juego de prueba para los caminos 0-1-3-4-5-7-8-9-12-13-15-14 y 0-1-3-4-5-7-8-9-12-13-15-16. Es decir, **el subprograma es incorrecto** por que contiene dos caminos que no pueden recorrerse. Concretamente, la imposibilidad de transitar por el camino 0-1-3-4-5-7-8-9-12-13-15-16 hace que el subprograma clasifique erróneamente los triángulos equiláteros como isósceles.

	Camino	Resultado esperado	Juego de prueba
1	0-1-2	Parámetros erróneos: valores no naturales	a=0, b=1, c=1
2	0-1-3-2		a=1, b=-3, c=1
3	0-1-3-4-2		a=3, b=1, c=0
4	0-1-3-4-5-6	Parámetros erróneos: imposible cerrar el triángulo	a=5, b=1, c=1
5	0-1-3-4-5-7-6		a=2, b=4, c=2
6	0-1-3-4-5-7-8-6		a=3, b=1, c=5
7	0-1-3-4-5-7-8-9-10	Triángulo isósceles	a=2, b=2, c=1
8	0-1-3-4-5-7-8-9-11-10		a=2, b=3, c=3
9	0-1-3-4-5-7-8-9-12-10		a=4, b=3, c=4
10	0-1-3-4-5-7-8-9-12-13-14	Triángulo escaleno	a=3, b=4, c=5
11	0-1-3-4-5-7-8-9-12-13-15-14		Es imposible crear un juego de prueba
12	0-1-3-4-5-7-8-9-12-13-15-16	Triángulo equilátero	

ESTE EJERCICIO ES DE TIPO MIXTO.

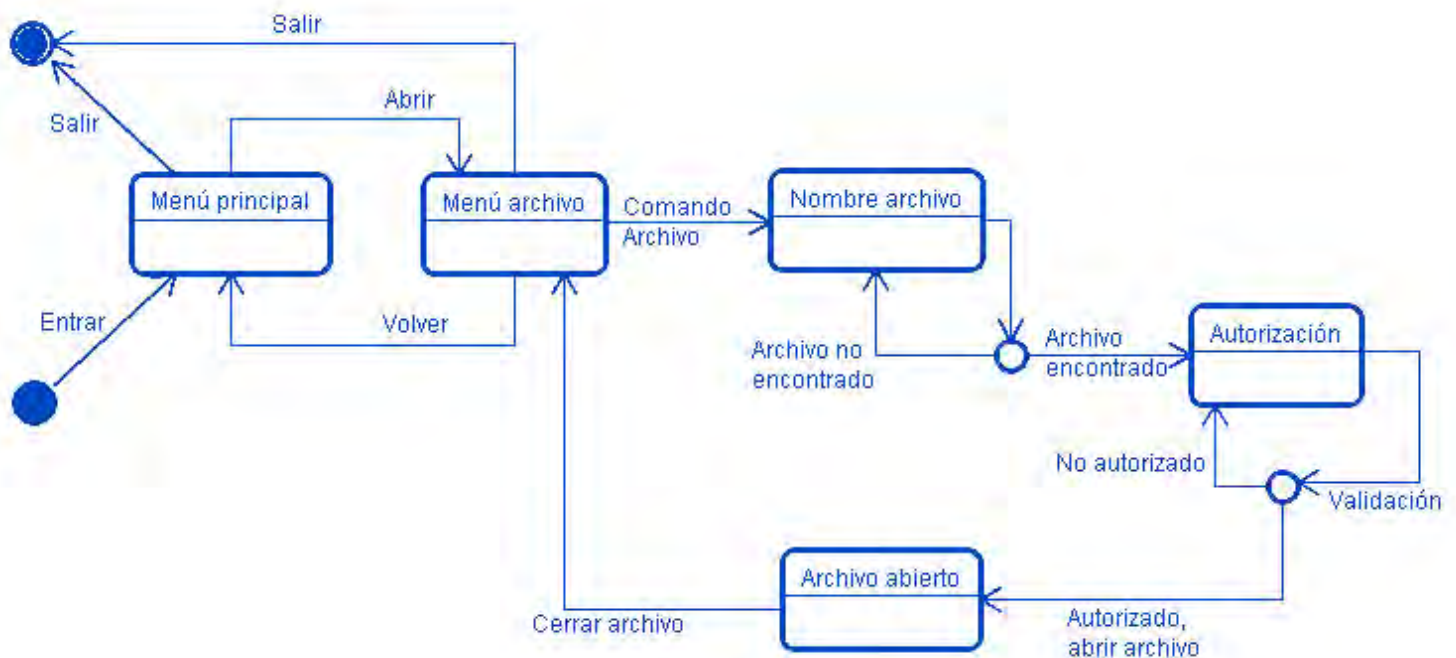
ES IRRELEVANTE SI CONTESTA A LA PREGUNTA DE TEST O NO. SIN EMBARGO, SE DEBE ESCANEAR DICHA HOJA JUNTO CON EL RESTO DE LA CONTESTACIÓN DEL EXAMEN. EL ALUMNO PUEDE QUEDARSE CON EL ENUNCIADO.

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. Una interfaz gráfica tiene un “Menú principal” con un submenú de “Archivo” que, a su vez, tiene el comando de “Abrir archivo”. Cada menú tiene una opción para “Salir”. El comando “Abrir archivo” tiene un cuadro de texto en el que se puede escribir el nombre (y opcionalmente la ruta) del archivo que se quiere abrir. Supóngase que sólo se puede abrir un archivo cada vez. Si el nombre del archivo es correcto, se pedirá que se escriba una clave para autorizar su apertura y, si es incorrecta, la interfaz vuelve al submenú “Archivo”. **Construya un modelo del comportamiento (para el análisis) utilizando un diagrama de transición de estados.**

Solución

2. Varios módulos de una aplicación utilizan la información de un fichero que está en un disco. Para usarlo, los módulos invocan los servicios que provee la controladora del dispositivo –los cuales trabajan directamente con las características físicas del disco—. Por ejemplo, para escribir en el fichero, los módulos invocan la instrucción imaginaria del microcontrolador del disco: `WRITE_STRING(DiskID, Cyl, Sector, Pos, NBytes, "String")`. ¿Qué tipo de acoplamiento presenta este caso? ¿Qué consecuencias tiene este uso respecto a la reusabilidad y a la mantenibilidad? ¿Cómo cambiaría esta situación?

Solución

Página 152 de bibliografía básica.

Éste es un claro ejemplo de acoplamiento externo, en el que varios módulos comparten información que reside en un dispositivo externo a ellos y a la que acceden directamente a través del ‘hardware’ que controla el dispositivo.

Esta práctica obliga a realizar todas las operaciones a través de las rutinas de servicio del propio dispositivo. Al ser, dichas rutinas, específicas del dispositivo; la posibilidad de reutilización del código –reusabilidad— en cualquier otro dispositivo que no sea idéntico, está –obviamente— disminuido o anulado. En cuanto a la mantenibilidad –o facilidad y posibilidad de hacer modificaciones en el código—, también se ve gravemente afectada; puesto que, si se cambia algo del dispositivo externo, obliga a cambiar todas las partes del código en las que se utilice dicho dispositivo.

La manera de evitar todo esto es mediante la abstracción y el encapsulamiento de todas estas operaciones en un módulo o paquete que se constituya como un controlador abstracto y genérico del dispositivo –disco— y que contenga, bajo invocaciones genéricas a los antiguos servicios específicos del dispositivo, las rutinas concretas de operación del dispositivo. Así, si se produce algún cambio, sólo será necesario hacer las modificaciones en el ‘controlador abstracto’.

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

3. Un fichero secuencial almacena información de entradas y salidas de stocks de un almacén. Estas transacciones están agrupadas en lotes. Cada lote consiste en una serie de transacciones referentes a un mismo producto. Los lotes comienzan con una marca de cabecera, un conjunto de registros con las entradas o salidas del producto y terminan con otra marca de fin.

Diseñe un programa que cree un informe con los movimientos netos de cada producto. Utilice la metodología de Jackson para las estructuras de datos de entrada y salida así como para la estructura del programa.

Solución

