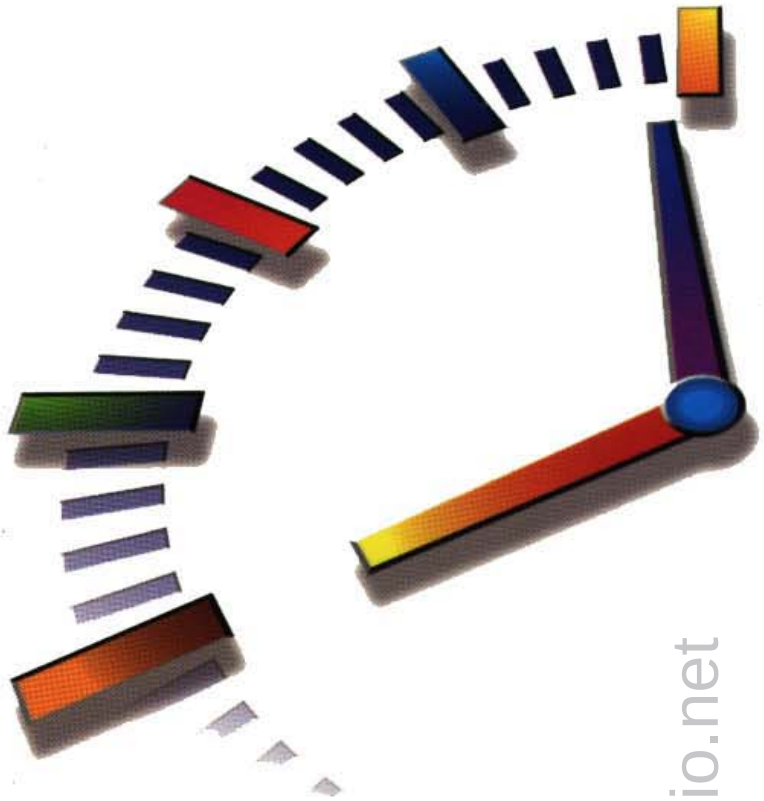
*Si sólo tiene tiempo para las respuestas*

*24 lecciones en las que invertirá una hora por lección*



# Aprendiendo® UML en 24 Horas

**Joseph Schmuller**

Prentice Hall

www.elsolucionario.net

www.elsolucionario.net

Find your solutions manual here!

El Solucionario
www.elsolucionario.net

Subscribe RSS
Find on Facebook
Follow my Tweets

*Encuentra en nuestra página los Textos Universitarios que necesitas!*

*Libros y Solucionarios en formato digital*
*El complemento ideal para estar preparados para los exámenes!*

*Los Solucionarios contienen TODOS los problemas del libro resueltos*
*y explicados paso a paso de forma clara..*

*Visitanos para descargarlos GRATIS!*
*Descargas directas mucho más fáciles...*

WWW.ELSOLUCIONARIO.NET

Biology Investigación Operativa Computer Science
Matemáticas Avanzadas
Physics Estadística Chemistry Geometría
Math
Termodinámica Cálculo Electrónica Circuitos Business
Mechanical Engineering
Civil Engineering Economía Análisis Numérico
Ecuaciones Diferenciales
Electrical Engineering Álgebra
Electromagnetismo

Find your solutions manual here!

# Resumen de contenido


| | | |
|---|---|---|
| | Introducción | 1 |
| **PARTE I PARA INICIAR** | | **3** |
| Hora 1 | Introducción al UML | 5 |
| 2 | Orientación a objetos | 19 |
| 3 | Uso de la orientación a objetos | 33 |
| 4 | Uso de relaciones | 45 |
| 5 | Agregación, composición, interfaces y realización | 57 |
| 6 | Introducción a los casos de uso | 67 |
| 7 | Diagramas de casos de uso | 75 |
| 8 | Diagramas de estados | 91 |
| 9 | Diagramas de secuencias | 103 |
| 10 | Diagramas de colaboraciones | 119 |
| 11 | Diagramas de actividades | 133 |
| 12 | Diagramas de componentes | 149 |
| 13 | Diagramas de distribución | 163 |
| 14 | Nociones de los fundamentos del UML | 173 |
| 15 | Adaptación del UML en un proceso de desarrollo | 187 |
| **PARTE II ESTUDIO DE UN CASO** | | **203** |
| Hora 16 | Presentación del caso por estudiar | 205 |
| 17 | Elaboración de un análisis de dominio | 223 |
| 18 | Recopilación de las necesidades del sistema | 247 |
| 19 | Desarrollo de los casos de uso | 267 |
| 20 | Orientación a las interacciones y cambios de estado | 281 |
| 21 | Diseño del aspecto, sensación y distribución | 293 |
| 22 | Noción de los patrones de diseño | 309 |


www.elsolucionario.net

www.elsolucionario.net

**PARTE III VISIÓN DEL FUTURO** **321**

Hora 23 Modelado de sistemas incrustados 323

24 El futuro del UML 341

**PARTE IV APÉNDICES** **355**

Apéndice A Respuestas a los cuestionarios 357

Apéndice B Herramientas de modelado para el UML 369

Apéndice C Un resumen gráfico 377

Índice 387

www.elsolucionario.net

www.elsolucionario.net

# Contenido


INTRODUCCIÓN 1

PARTE I PARA INICIAR 3

HORA 1 INTRODUCCIÓN AL UML 5

Por qué es necesario el UML ....6
La concepción del UML ....7
Diagramas del UML ....8
Diagrama de clases ....8
Diagrama de objetos ....9
Diagrama de casos de uso ....10
Diagrama de estados ....10
Diagrama de secuencias ....11
Diagrama de actividades ....12
Diagrama de colaboraciones ....13
Diagrama de componentes ....13
Diagrama de distribución ....14
Otras características ....14
Paquetes ....14
Notas ....15
Estereotipos ....15
Para qué tantos diagramas ....16
Resumen ....16
Preguntas y respuestas ....16
Taller ....17
Cuestionario ....17
Ejercicios ....17

HORA 2 ORIENTACIÓN A OBJETOS 19

Objetos, objetos por doquier ....20
Algunos conceptos ....22
Abstracción ....22
Herencia ....23
Polimorfismo ....24
Encapsulamiento ....25
Envío de mensajes ....26
Asociaciones ....26
Agregación ....28
La recompensa ....30


www.elsolucionario.net

www.elsolucionario.net

Resumen ....30
Preguntas y respuestas ....31
Taller ....31
Cuestionario ....31
Ejercicios ....32

**HORA 3 USO DE LA ORIENTACIÓN A OBJETOS** **33**

Concepción de una clase ....33
Atributos ....34
Operaciones ....36
Atributos, operaciones y concepción ....37
Responsabilidades y restricciones ....38
Notas adjuntas ....40
Qué es lo que hacen las clases y cómo encontrarlas ....40
Resumen ....43
Preguntas y respuestas ....43
Taller ....44
Cuestionario ....44
Ejercicios ....44

**HORA 4 USO DE RELACIONES** **45**

Asociaciones ....46
Restricciones en las asociaciones ....47
Clases de asociación ....48
Vínculos ....48
Multiplicidad ....49
Asociaciones calificadas ....50
Asociaciones reflexivas ....51
Herencia y generalización ....51
Descubrimiento de la herencia ....53
Clases abstractas ....53
Dependencias ....54
Resumen ....55
Preguntas y respuestas ....55
Taller ....56
Cuestionarios ....56
Ejercicios ....56

**HORA 5 AGREGACIÓN, COMPOSICIÓN, INTERFACES Y REALIZACIÓN** **57**

Agregaciones ....58
Restricciones en las agregaciones ....59
Composiciones ....59
Contextos ....59

www.elsolucionario.net

www.elsolucionario.net

Interfaces y realizaciones ....61
Visibilidad ....62
Ámbito ....63
Resumen ....63
Preguntas y respuestas ....64
Taller ....64
Cuestionario ....64
Ejercicios ....64

**HORA 6 INTRODUCCIÓN A LOS CASOS DE USO 67**

Qué son los casos de uso ....68
Importancia de los casos de uso ....69
Un ejemplo: la máquina de gaseosas ....69
El caso de uso "Comprar gaseosa" ....70
Casos de uso adicionales ....71
Inclusión de los casos de uso ....72
Extensión de los casos de uso ....72
Inicio del análisis de un caso de uso ....73
Resumen ....73
Preguntas y respuestas ....74
Taller ....74
Cuestionario ....74
Ejercicios ....74

**HORA 7 DIAGRAMAS DE CASOS DE USO 75**

Representación de un modelo de caso de uso ....76
Una nueva visita a la máquina de gaseosas ....76
Secuencia de pasos en los escenarios ....77
Concepción de las relaciones entre casos de uso ....78
Inclusión ....78
Extensión ....79
Generalización ....80
Agrupamiento ....81
Diagramas de casos de uso en el proceso de análisis ....81
Aplicación de los modelos de caso de uso ....81
Comprensión del dominio ....82
Comprensión de los usuarios ....82
Comprensión de los casos de uso ....82
Profundización ....83
Dónde estamos ....85
Elementos estructurales ....86
Relaciones ....86
Agrupamiento ....86

www.elsolucionario.net

www.elsolucionario.net

Anotación ....86
Extensión ....87
...y más ....87
El Panorama ....87
Resumen ....88
Preguntas y respuestas ....88
Taller ....89
Cuestionario ....89
Ejercicios ....89

**HORA 8 DIAGRAMAS DE ESTADOS** **91**

Qué es un diagrama de estados ....92
Simbología ....92
Adición de detalles al icono de estado ....93
Sucesos y acciones ....94
Condiciones de seguridad ....95
Subestados ....96
Subestados secuenciales ....96
Subestados concurrentes ....96
Estados históricos ....97
Mensajes y señales ....98
Por qué son importantes los diagramas de estados ....99
Adiciones al panorama ....99
Resumen ....100
Preguntas y respuestas ....101
Taller ....101
Cuestionarios ....101
Ejercicios ....102

**HORA 9 DIAGRAMAS DE SECUENCIAS** **103**

Qué es un diagrama de secuencias ....104
Objetos ....104
Mensaje ....104
Tiempo ....105
La GUI ....106
La secuencia ....106
El diagrama de secuencias ....106
El caso de uso ....108
Instancias y genéricos ....108
Un diagrama de secuencias de instancias ....108
Un diagrama de secuencias genérico ....109
Creación de un objeto en la secuencia ....112
Cómo representar la recursividad ....114

www.elsolucionario.net

www.elsolucionario.net

Adiciones al panorama ....115
Resumen ....115
Preguntas y respuestas ....116
Taller ....116
Cuestionario ....116
Ejercicios ....117

**HORA 10 DIAGRAMAS DE COLABORACIONES 119**

Qué es un diagrama de colaboraciones ....120
La GUI ....121
Cambios de estado ....122
La máquina de gaseosas ....122
Creación de un objeto ....124
Algunos conceptos más ....125
Varios objetos receptores en una clase ....126
Representación de los resultados ....126
Objetos activos ....127
Sincronización ....127
Adiciones al panorama ....128
Resumen ....129
Preguntas y respuestas ....130
Taller ....130
Cuestionario ....130
Ejercicios ....130

**HORA 11 DIAGRAMAS DE ACTIVIDADES 133**

Objetivos ....133
Qué es un diagrama de actividades ....134
Decisiones, decisiones, decisiones ....135
Rutas concurrentes ....135
Indicaciones ....136
Aplicación de los diagramas de actividades ....137
Una operación: Fibs ....137
Proceso de creación de un documento ....138
Marcos de responsabilidad ....140
Diagramas híbridos ....142
Adiciones al panorama ....144
Resumen ....145
Preguntas y respuestas ....146
Taller ....146
Cuestionario ....146
Ejercicios ....147

www.elsolucionario.net

www.elsolucionario.net

**HORA 12 DIAGRAMAS DE COMPONENTES** **149**

Qué es un componente ....150
Componentes e interfaces ....150
Sustitución y reutilización ....151
Tipos de componentes ....152
Qué es un diagrama de componentes ....152
Representación de un componente ....152
Cómo representar las interfaces ....153
Aplicación de los diagramas de componentes ....154
Una página Web con un subprograma Java ....154
Una página Web con controles ActiveX ....156
PowerToys ....157
Diagramas de componentes en el panorama ....158
Resumen ....159
Preguntas y respuestas ....160
Taller ....160
Cuestionario ....160
Ejercicios ....160

**HORA 13 DIAGRAMAS DE DISTRIBUCIÓN** **163**

Qué es un diagrama de distribución ....164
Aplicación de los diagramas de distribución ....165
Un equipo doméstico ....166
Una red token-ring ....166
ARCnet ....167
Thin ethernet ....168
Red inalámbrica Ricochet de Metricom ....169
Los diagramas de distribución en el panorama ....170
Resumen ....171
Preguntas y respuestas ....172
Taller ....172
Cuestionario ....172
Ejercicios ....172

**HORA 14 NOCIONES DE LOS FUNDAMENTOS DEL UML** **173**

Estructura del UML ....174
Capa del metamodelado: cercano y personal ....175
El paquete de Fundamentos ....176
El paquete de los elementos de comportamiento ....178
Administración de modelos ....179
Extensión del UML ....179

www.elsolucionario.net

www.elsolucionario.net

Estereotipos ....179
Dependencia ....180
Clasificador ....180
Clase ....181
Generalización ....181
Paquete ....181
Componente ....182
Algunos otros estereotipos ....182
Estereotipos gráficos ....182
Restricciones ....183
Valores etiquetados ....184
Resumen ....184
Preguntas y respuestas ....185
Taller ....185
Cuestionario ....185

**HORA 15 ADAPTACIÓN DEL UML EN UN PROCESO DE DESARROLLO 187**

Metodologías: antiguas y recientes ....188
El método antiguo ....188
El método reciente ....189
Lo que debe hacer un proceso de desarrollo ....190
GRAPPLE ....191
RAD$^3$: la estructura de GRAPPLE ....192
Recopilación de necesidades ....193
Análisis ....195
Diseño ....197
Desarrollo ....198
Distribución ....199
Resumen de GRAPPLE ....199
Resumen ....200
Preguntas y respuestas ....201
Taller ....201
Cuestionario ....201

**PARTE II ESTUDIO DE UN CASO 203**

**HORA 16 PRESENTACIÓN DEL CASO POR ESTUDIAR 205**

Aplicación de GRAPPLE al problema ....206
Descubrir los procesos del negocio ....206
Servir a un cliente ....207
Preparación de platillos ....215
Limpieza de la mesa ....218
Lecciones aprendidas ....219

www.elsolucionario.net

www.elsolucionario.net

Resumen ....220
Preguntas y respuestas ....221
Taller ....221
Cuestionario ....221
Ejercicios ....222

**HORA 17 ELABORACIÓN DE UN ANÁLISIS DE DOMINIO 223**

Análisis de la entrevista del proceso del negocio ....224
Desarrollo del diagrama de clases inicial ....225
Agrupación de las clases ....228
Conformación de asociaciones ....229
Asociaciones con el cliente ....229
Asociaciones con el Mesero ....234
Asociaciones con el Chef ....235
Asociaciones con el Mozo de piso ....236
Asociaciones con el Gerente ....236
Una digresión ....237
Formación de agregados y objetos compuestos ....238
Llenado de las clases ....239
El Cliente ....240
El Empleado ....240
La Cuenta ....242
Detalles generales de los modelos ....242
Diccionario del modelo ....242
Organización del diagrama ....243
Lecciones aprendidas ....243
Resumen ....244
Preguntas y respuestas ....244
Taller ....244
Cuestionario ....244
Ejercicios ....245

**HORA 18 RECOPILACIÓN DE LAS NECESIDADES DEL SISTEMA 247**

Desarrollo de la idea ....248
Preparación para la recopilación de las necesidades ....257
La sesión JAD de necesidades ....258
El resultado ....261
¿Ahora qué? ....264
Resumen ....264
Preguntas y respuestas ....265
Taller ....265
Cuestionario ....265
Ejercicio ....265

www.elsolucionario.net

www.elsolucionario.net

**HORA 19 DESARROLLO DE LOS CASOS DE USO** **267**

Cuidado y provisión de los casos de uso ....268
El análisis de los casos de uso ....268
El paquete Mesero ....269
Tomar una orden ....270
Transmitir la orden a la cocina ....271
Cambiar una orden ....272
Sondeo del progreso de la orden ....273
Notificar al chef del progreso de los clientes en sus alimentos ....273
Totalizar una cuenta ....275
Imprimir una Cuenta ....275
Llamar a un Asistente ....276
Casos de uso restantes ....277
Componentes del sistema ....277
Resumen ....278
Preguntas y respuestas ....278
Taller ....279
Cuestionario ....279
Ejercicios ....279

**HORA 20 ORIENTACIÓN A LAS INTERACCIONES Y CAMBIOS DE ESTADO** **281**

Las partes funcionales del sistema ....282
El paquete Mesero ....282
El paquete Chef ....283
El paquete Mozo De Piso ....283
El paquete Asistente Mesero ....283
El paquete Asistente Chef ....283
El paquete Cantinero ....284
El paquete Encargado Del Guardarropa ....284
Colaboración en el sistema ....284
Tomar una orden ....285
Cambiar una orden ....288
Sondeo del progreso de la orden ....289
Implicaciones ....290
Resumen ....291
Preguntas y respuestas ....291
Taller ....292
Cuestionario ....292
Ejercicios ....292

www.elsolucionario.net

www.elsolucionario.net

**HORA 21 DISEÑO DEL ASPECTO, SENSACIÓN Y DISTRIBUCIÓN** **293**

Algunos principios generales en el diseño de las GUI ....294
La sesión JAD para la GUI ....296
De los casos de uso a las interfaces de usuario ....297
Diagramas UML para el diseño de la GUI ....299
Esbozos de la distribución del sistema ....300
La red ....301
Los nodos y el diagrama de distribución ....301
Siguientes pasos ....303
...Y ahora, unas palabras de nuestros patrocinadores ....304
Mejorar el trabajo de la fuerza de ventas ....304
Expansiones en el mundo restaurantero ....305
Resumen ....306
Preguntas y respuestas ....307
Taller ....308
Cuestionario ....308
Ejercicios ....308

**HORA 22 NOCIÓN DE LOS PATRONES DE DISEÑO** **309**

Parametrización ....310
Patrones de diseño ....312
Cadena de responsabilidad ....313
Cadena de responsabilidad: dominio Restaurante ....314
Cadena de responsabilidad: Modelos de eventos de los exploradores Web 315
Nuestros propios patrones de diseño ....317
Ventajas de los patrones de diseño ....319
Resumen ....319
Preguntas y respuestas ....320
Taller ....320
Cuestionario ....320
Ejercicios ....320

**PARTE III VISIÓN DEL FUTURO** **321**

**HORA 23 MODELADO DE SISTEMAS INCRUSTADOS** **323**

La madre de la invención ....324
Creación de TecnoApretón ....325
¿Qué es un sistema incrustado? ....327
Conceptos de los sistemas incrustados ....328
Tiempo ....328
Subprocesos ....328
Interrupciones ....329
Sistema operativo ....330

www.elsolucionario.net

www.elsolucionario.net

Modelado de TecnoApretón ....332
Clases ....332
Casos de uso ....334
Interacciones ....335
Cambios de estado generales ....337
Distribución ....338
Flexiones en sus músculos ....338
Resumen ....339
Preguntas y respuestas ....339
Taller ....340
Cuestionario ....340
Ejercicios ....340

**HORA 24 EL FUTURO DEL UML 341**

Extensiones para los negocios ....342
Lecciones de las extensiones de negocios ....343
Interfaces gráficas de usuario ....343
Conexiones a casos de uso ....344
Modelado de la GUI ....344
Sistemas expertos ....346
Componentes de un sistema experto ....346
Un ejemplo ....348
Modelado de la base de conocimientos ....349
Eso es todo, amigos ....352
Resumen ....352
Preguntas y respuestas ....353
Taller ....353
Cuestionario ....353

**PARTE IV APÉNDICES 355**

**APÉNDICE A RESPUESTAS A LOS CUESTIONARIOS 357**

**APÉNDICE B HERRAMIENTAS DE MODELADO PARA EL UML 369**

Características en común ....369
Rational Rose ....370
SELECT Enterprise ....372
Visual UML ....374
La herramienta ideal para el modelado ....375

www.elsolucionario.net

www.elsolucionario.net

**APÉNDICE C UN RESUMEN GRÁFICO** 377

Diagrama de actividades ....378
Diagrama de clases ....380
Diagrama de colaboraciones ....382
Diagrama de componentes ....382
Diagrama de distribución ....383
Diagrama de secuencias ....383
Diagrama de estados ....384
Diagrama de casos de uso ....385

**ÍNDICE** 387

www.elsolucionario.net

www.elsolucionario.net

# Acerca del autor

**Joseph Schmuller** es vicepresidente de la división de Consumer Finance Technologies del Bank of America. De 1991 a 1997 fue editor en jefe de la revista PC AI. Ha escrito diversos artículos y reseñas de tecnologías avanzadas de computación y es autor de *ActiveX No experience required* y *Dynamic HTML Master the Essentials*. Tiene un doctorado de la Universidad de Wisconsin, y es profesor adjunto en la Universidad del Norte de Florida.

www.elsolucionario.net

www.elsolucionario.net

# Dedicatoria

*A mi maravillosa madre, Sara Riba Schmuller,*
*quien me enseñó a aprender por mí mismo.*

# Reconocimientos

Escribir un libro es un proceso arduo; pero por fortuna, el equipo de Macmillan Computer Publishing lo ha hecho más fácil. Es un placer reconocer sus contribuciones. Tanto el editor de adquisiciones, Chris Webb, como el de Desarrollo, Matt Purcell, me ayudaron a convertir mis pensamientos en algo legible; por encima de su gran experiencia editorial, les agradezco sus alicientes, paciencia y apoyo. Los revisores técnicos, Bill Rowe y Michael Tobler se aseguraron de que el contenido fuera técnicamente correcto y se los agradezco. La editora, Susan Moore, los destacados artistas de Macmillan y el personal de producción convirtieron el manuscrito y sus diversos diagramas en el libro que ahora está leyendo.

David Fugate de Waterside Productions conjugó todo el proceso. Le agradezco haberme hecho coincidir con Macmillan y haberme colocado en otro proyecto muy retribuyente.

Tengo el privilegio de trabajar todos los días con un grupo de excelentes profesionales en la división de Consumer Finance Technologies del Bank of America (específicamente, como miembro del grupo de Objetos y componentes reutilizables). Mi agradecimiento a mis colegas por su apoyo y cooperación. En particular, las conversaciones con Keith Barret y Rob Warner me ayudaron a clarificar mis ideas sobre diversos puntos. Por desgracia Tom Williamson, nuestro Director de división, falleció mientras escribía este libro. Él era el corazón y el alma de CFT, y fue un asesor, tutor, colega y amigo.

Agradezco a mis queridos amigos, los Spragues de Madison, Wisconsin, en cuyo vecindario estaba de casualidad cuando empecé a escribir este libro y, nuevamente, al terminarlo. Agradezco a mi madre y a mi hermano David por su amor y por siempre estar cerca de mí, y a Kathryn por ser, por siempre, todo para mí.

www.elsolucionario.net

www.elsolucionario.net

# Pearson Educación Latinoamérica

El personal de Pearson Educación Latinoamérica está comprometido en presentarle lo mejor en material de consulta sobre computación. Cada libro de Pearson Educación Latinoamérica es el resultado de meses de trabajo de nuestro personal, que investiga y refina la información que se ofrece.

Como parte de este compromiso con usted, el lector de Pearson Educación Latinoamérica lo invita a dar su opinión. Por favor háganos saber si disfruta este libro, si tiene alguna dificultad con la información y los ejemplos que se presentan, o si tiene alguna sugerencia para la próxima edición.

Sin embargo, recuerde que el personal de Pearson Educación Latinoamérica no puede actuar como soporte técnico o ni responder preguntas acerca de problemas relacionados con el software o el hardware.

Si usted tiene alguna pregunta o comentario acerca de cualquier libro de Pearson Educación Latinoamérica, existen muchas formas de entrar en contacto con nosotros. Responderemos a todos los lectores que podamos. Su nombre, dirección y número telefónico jamás formarán parte de ninguna lista de correos ni serán usados para otro fin, más que el de ayudarnos a seguirle llevando los mejores libros posibles. Puede escribirnos a la siguiente dirección:

Pearson Educación Latinoamérica

Attn: Editorial División Computación

Calle Cuatro No. 25, 2º Piso,

Col. Fracc. Alce Blanco

Naucalpan de Juárez, Edo. de México.

C.P. 53370

Si lo prefiere, puede mandar un fax a Pearson Educación Latinoamérica al (525) 5387-0811.

También puede ponerse en contacto con Pearson Educación Latinoamérica a través de nuestra página Web: `http://www.pearson.com.mx`

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# Introducción

Todo gira en torno de una visión. Un sistema complejo toma forma cuando alguien tiene la visión de cómo la tecnología puede mejorar las cosas. Los desarrolladores tienen que entender completamente la idea y mantenerla en mente mientras crean el sistema que le dé foma.

El éxito de los proyectos de desarrollo de aplicaciones o sistemas se debe a que sirven como enlace entre quien tiene la idea y el desarrollador. El UML (*Lenguaje Unificado de Modelado*) es una herramienta que cumple con esta función, ya que le ayuda a capturar la idea de un sistema para comunicarla posteriormente a quien esté involucrado en su proceso de desarrollo; esto se lleva a cabo mediante un conjunto de símbolos y diagramas. Cada diagrama tiene fines distintos dentro del proceso de desarrollo.

El objetivo de este libro es darle, en 24 horas de estudio, las bases sobre el UML. Cada hora le presentará ejemplos para mejorar la comprensión e incluirá ejercicios que le permitirán practicar sus recién adquiridos conocimientos.

Dividí este libro en tres partes: la primera parte le da un panorama del UML y le explica la orientación a objetos, misma que conforma los conceptos fundamentales de la diagramación de objetos y clases. Examinaremos los casos de uso (una estructura para mostrar la forma en que un sistema lucirá ante el usuario, para luego mostrar cómo hacer diagramas de esta estructura). Las horas restantes de la primera parte le permitirán trabajar con el resto de los diagramas UML.

La segunda parte le muestra una metodología simplificada para el desarrollo, enriquecida con el estudio de un caso ficticio. Así, las horas de la segunda parte le mostrarán la forma en que el UML se adapta al contexto de un proyecto de desarrollo. Verá la forma en que los elementos del UML funcionan en conjunto para modelar un sistema.

Por último, en la tercera parte aplicaremos el UML para diseñar patrones y sistemas incrustados, y examinaremos su campo de aplicación en dos áreas más.

Existen diversos fabricantes que cuentan con paquetes que le permitirán generar diagramas UML y coordinarlos en un modelo. Los más notables son Rational Rose y SELECT Enterprise, aunque cabe mencionar que Visual UML es otro digno contendiente. Microsoft está autorizado para utilizar la tecnología de Rational y así comercializa Visual Modeler, un subconjunto de Rational Rose. No obstante, en este libro todo lo que necesitará será un lápiz y papel para dibujar los diagramas y una sana curiosidad respecto al estado actual del diseño de sistemas.

¡Iniciemos!

www.elsolucionario.net

www.elsolucionario.net

# Convenciones utilizadas en este libro

En los diagramas incluidos en este libro no utilizamos vocales con acento, ni la letra eñe. Esto debido a que el alfabeto inglés, de donde procede la mayoría de los lenguajes de programación, no los incluye y el empleo de esos caracteres en sus diagramas podría acarrearle problemas tanto en el UML como en el lenguaje de programación que piense

Una Nota presenta interesantes secciones de información relacionadas con el tema que se trate.

TÉRMINO NUEVO

El icono Término Nuevo resalta las definiciones de vocablos nuevos y esenciales. El vocablo, en sí, aparecerá en *cursiva*.

www.elsolucionario.net

www.elsolucionario.net

# PARTE I
# Para iniciar

## Hora


1 Introducción al UML
2 Orientación a objetos
3 Uso de la orientación a objetos
4 Uso de relaciones
5 Agregación, composición, interfaces y realizaci
6 Introducción a los casos de uso
7 Diagramas de casos de uso
8 Diagramas de estados
9 Diagramas de secuencias
10 Diagramas de colaboraciones
11 Diagramas de actividades
12 Diagramas de componentes
13 Diagramas de distribución
14 Nociones de los fundamentos del UML
15 Adaptación del UML en un proceso de desarro


www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 1

# Introducción al UML

El UML (Lenguaje Unificado de Modelado) es una de las herramientas más emocionantes en el mundo actual del desarrollo de sistemas. Esto se debe a que permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas.

En esta hora se tratarán los siguientes temas:

- Por qué es necesario el UML
- La concepción del UML
- Diagramas del UML
- Para qué tantos diagramas

**TÉRMINO NUEVO** En el contexto de este libro considere a un *sistema* como una combinación de software y hardware que da una solución a un problema de negocios. El *desarrollo de sistemas* es la creación de un programa para un *cliente*, este último es quien tiene el problema que debe ser resuelto. Un *analista* es el que documenta el problema del cliente y lo comunica a los *desarrolladores*, que son los programadores que generarán el programa que resolverá el problema y lo distribuirán en equipos de computación.

www.elsolucionario.net

www.elsolucionario.net

La comunicación de la idea es de suma importancia. Antes del advenimiento del UML, el desarrollo de sistemas era, con frecuencia, una propuesta al azar. Los analistas de sistemas intentaban evaluar los requerimientos de sus clientes, generar un análisis de requerimientos en algún tipo de notación que ellos mismos comprendieran (aunque el cliente no lo comprendiera), dar tal análisis a uno o varios programadores y esperar que el producto final cumpliese con lo que el cliente deseaba.

Dado que el desarrollo de sistemas es una actividad humana, hay muchas posibilidades de cometer errores en cualquier etapa del proceso, por ejemplo, el analista pudo haber malentendido al cliente, es decir, probablemente produjo un documento que el cliente no pudo comprender. Tal vez ese documento tampoco fue comprendido por los programadores quienes, por ende, pudieron generar un programa difícil de utilizar y no generar una solución al problema original del cliente.

¿Alguien se preguntará por qué muchos de los sistemas en uso son ineficientes, engorrosos y difíciles de utilizar?

# Por qué es necesario el UML

En los principios de la computación, los programadores no realizaban análisis muy profundos sobre el problema por resolver. Si acaso, garabateaban algo en una servilleta. Con frecuencia comenzaban a escribir el programa desde el principio, y el código necesario se escribía conforme se requería. Aunque anteriormente esto agregaba un aura de aventura y atrevimiento al proceso, en la actualidad es inapropiado en los negocios de alto riesgo.

Hoy en día, es necesario contar con un plan bien analizado. Un cliente tiene que comprender qué es lo que hará un equipo de desarrolladores; además tiene que ser capaz de señalar cambios si no se han captado claramente sus necesidades (o si cambia de opinión durante el proceso). A su vez, el desarrollo es un esfuerzo orientado a equipos, por lo que cada uno de sus miembros tiene que saber qué lugar toma su trabajo en la solución final (así como saber cuál es la solución en general).

Conforme aumenta la complejidad del mundo, los sistemas informáticos también deberán crecer en complejidad. En ellos se encuentran diversas piezas de hardware y software que se comunican a grandes distancias mediante una red, misma que está vinculada a bases de datos que, a su vez, contienen enormes cantidades de información. Si desea crear sistemas que lo involucren con este nuevo milenio ¿cómo manejará tanta complejidad?

La clave está en organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan y convengan con él. El UML proporciona tal organización.

Un arquitecto no podría crear una compleja estructura como lo es un edificio de oficinas sin crear primero un anteproyecto detallado; asimismo usted tampoco podría generar un complejo sistema en un edificio de oficinas sin crear un plan de diseño detallado. La idea

www.elsolucionario.net

www.elsolucionario.net

es que así como un arquitecto le muestra un anteproyecto a la persona que lo contrató, usted deberá mostrarle su plan de diseño al cliente. Tal plan de diseño debe ser el resultado de un cuidadoso análisis de las necesidades del cliente.

Otra característica del desarrollo de sistemas contemporáneo es reducir el periodo de desarrollo. Cuando los plazos se encuentran muy cerca uno del otro es absolutamente necesario contar con un diseño sólido.

Hay otro aspecto de la vida moderna que demanda un diseño sólido: las adquisiciones corporativas. Cuando una empresa adquiere a otra, la nueva organización debe tener la posibilidad de modificar aspectos importantes de un proyecto de desarrollo que esté en progreso (la herramienta de desarrollo, el lenguaje de codificación, y otras cosas). Un anteproyecto bien diseñado facilitará la conversión. Si el diseño es sólido, un cambio en la implementación procederá sin problemas.

La necesidad de diseños sólidos ha traído consigo la creación de una notación de diseño que los analistas, desarrolladores y clientes acepten como pauta (tal como la notación en los diagramas esquemáticos sirve como pauta para los trabajadores especializados en electrónica). El UML es esa misma notación.

# La concepción del UML

El UML es la creación de Grady Booch, James Rumbaugh e Ivar Jacobson. Estos caballeros, apodados recientemente "Los tres amigos", trabajaban en empresas distintas durante la década de los años ochenta y principios de los noventa y cada uno diseñó su propia metodología para el análisis y diseño orientado a objetos. Sus metodologías predominaron sobre las de sus competidores. A mediados de los años noventa empezaron a intercambiar ideas entre sí y decidieron desarrollar su trabajo en conjunto.

Las horas 2, "Orientación a objetos", y 4, "Uso de relaciones", tratan de la orientación a objetos. Los conceptos de orientación a objetos tienen un papel fundamental en el desarrollo de este libro.

En 1994 Rumbaugh ingresó a Rational Software Corporation, donde ya trabajaba Booch. Jacobson ingresó a Rational un año después; el resto, como dicen, es historia.

Los anteproyectos del UML empezaron a circular en la industria del software y las reacciones resultantes trajeron consigo considerables modificaciones. Conforme diversos corporativos vieron que el UML era útil a sus propósitos, se conformó un consorcio del UML. Entre los miembros se encuentran DEC, Hewlett-Packard, Intellicorp, Microsoft, Oracle, Texas Instruments y Rational. En 1997 el consorcio produjo la versión 1.0 del UML y lo puso a consideración del OMG (Grupo de administración de objetos) como respuesta a su propuesta para un lenguaje de modelado estándar.

www.elsolucionario.net

www.elsolucionario.net

El consorcio aumentó y generó la versión 1.1, misma que se puso nuevamente a consideración del OMG. El grupo adoptó esta versión a finales de 1997. El OMG se encargó de la conservación del UML y produjo otras dos revisiones en 1998. El UML ha llegado a ser el estándar de facto en la industria del software, y su evolución continúa.

# Diagramas del UML

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos. En lugar de indicarle a usted cuáles son los elementos y las reglas, veamos directamente los diagramas ya que los utilizará para hacer el análisis del sistema.

Este enfoque es similar a aprender un idioma extranjero mediante el uso del mismo, en lugar de aprender sus reglas gramaticales y la conjugación de sus verbos. Después de un tiempo de hablar otro idioma se le facilitará la conjugación de verbos y la comprensión de las reglas gramaticales.

TÉRMINO NUEVO

La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como *modelo*. El modelo UML de un sistema es similar a un modelo a escala de un edificio junto con la interpretación del artista del edificio. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

A continuación se describirán brevemente los diagramas más comunes del UML y los conceptos que representan. Posteriormente, en la parte I verá cada uno de los diagramas con mayor detenimiento. Recuerde que es posible generar híbridos de estos diagramas y que el UML otorga formas de organizarlos y extenderlos.

## Diagrama de clases

Piense en las cosas que le rodean (una idea demasiado amplia, pero ¡inténtelo de cualquier forma!). Es probable que muchas de esas cosas tengan atributos (propiedades) y que realicen determinadas acciones. Podríamos imaginar cada una de esas acciones como un conjunto de tareas.

TÉRMINO NUEVO

También se encontrará con que las cosas naturalmente se albergan en categorías (automóviles, mobiliario, lavadoras...). A tales categorías las llamaremos clases. Una *clase* es una categoría o grupo de cosas que tienen atributos y acciones similares. He aquí un ejemplo: cualquier cosa dentro de la clase Lavadoras tiene atributos como son la marca, el modelo, el número de serie y la capacidad. Entre las acciones

www.elsolucionario.net

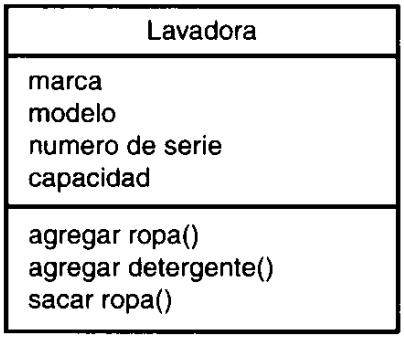
www.elsolucionario.net

de las cosas de esta clase se encuentran: "agregar ropa", "agregar detergente", "activarse" y "sacar ropa".

La figura 1.1 le muestra un ejemplo de la notación del UML que captura los atributos y acciones de una lavadora. Un rectángulo es el símbolo que representa a la clase, y se divide en tres áreas. El área superior contiene el nombre, el área central contiene los atributos, y el área inferior las acciones. Un diagrama de clases está formado por varios rectángulos de este tipo conectados por líneas que muestran la manera en que las clases se relacionan entre sí.

**FIGURA 1.1**

*El símbolo UML de una clase.*




¿Qué objetivo tiene pensar en las clases, así como sus atributos y acciones? Para interactuar con nuestro complejo mundo, la mayoría del software moderno simula algún aspecto del mundo. Décadas de experiencia sugieren que es más sencillo desarrollar aplicaciones que simulen algún aspecto del mundo cuando el software representa clases de cosas reales. Los diagramas de clases facilitan las representaciones a partir de las cuales los desarrolladores podrán trabajar.

A su vez, los diagramas de clases colaboran en lo referente al análisis. Permiten al analista hablarle a los clientes en su propia terminología, lo cual hace posible que los clientes indiquen importantes detalles de los problemas que requieren ser resueltos.

www.elsolucionario.net

## Diagrama de objetos

TÉRMINO NUEVO

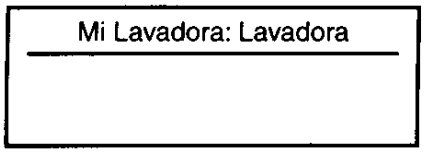
Un objeto es una instancia de clase (una entidad que tiene valores específicos de los atributos y acciones). Su lavadora, por ejemplo, podría tener la marca Laundatorium, el modelo Washmeister, el número de serie GL57774 y una capacidad de 7 Kg.

La figura 1.2 le muestra la forma en que el UML representa a un objeto. Vea que el símbolo es un rectángulo, como en una clase, pero el nombre está subrayado. El nombre de la instancia específica se encuentra a la izquierda de los dos puntos (:), y el nombre de la clase a la derecha.

www.elsolucionario.net

**FIGURA 1.2**

*El símbolo UML del objeto.*



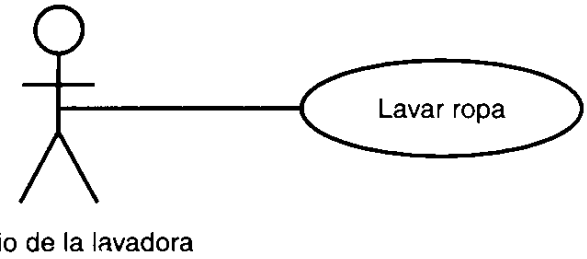

## Diagrama de casos de uso

**TÉRMINO NUEVO** Un caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario. Para los desarrolladores del sistema, ésta es una herramienta valiosa, ya que es una técnica de aciertos y errores para obtener los requerimientos del sistema desde el punto de vista del usuario. Esto es importante si la finalidad es crear un sistema que pueda ser utilizado por la gente en general (no sólo por expertos en computación).

Posteriormente trataremos este tema con mayor detalle; por ahora, le mostraré un ejemplo sencillo. Usted utiliza una lavadora, obviamente, para lavar su ropa. La figura 1.3 le muestra cómo representaría esto en un diagrama de casos de uso UML.

**FIGURA 1.3**

*Diagrama de casos de uso UML.*




**TÉRMINO NUEVO** A la figura correspondiente al Usuario de la lavadora se le conoce como actor. La elipse representa el caso de uso. Vea que el actor (la entidad que inicia el caso de uso) puede ser una persona u otro sistema.

## Diagrama de estados

En cualquier momento, un objeto se encuentra en un estado en particular. Una persona puede ser recién nacida, infante, adolescente, joven o adulta. Un elevador se moverá hacia arriba, estará en estado de reposo o se moverá hacia abajo. Una lavadora podrá estar en la fase de remojo, lavado, enjuague, centrifugado o apagada.

El diagrama de estados UML, que aparece en la figura 1.4, captura esta pequeña realidad. La figura muestra las transiciones de la lavadora de un estado al otro.

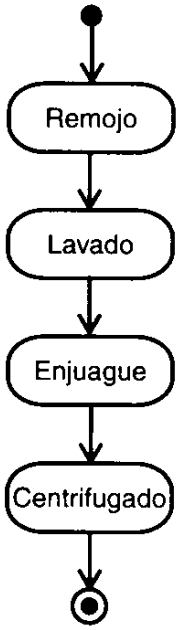
El símbolo que está en la parte superior de la figura representa el estado inicial y el de la parte inferior el estado final.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 1.4**

*Diagrama de estados UML.*




## Diagrama de secuencias

Los diagramas de clases y los de objeto representan información estática. No obstante, en un sistema funcional los objetos interactúan entre sí, y tales interacciones suceden con el tiempo. El diagrama de secuencias UML muestra la mecánica de la interacción con base en tiempos.

Continuando con el ejemplo de la lavadora, entre los componentes de la lavadora se encuentran: una manguera de agua (para obtener agua fresca), un tambor (donde se coloca la ropa) y un sistema de drenaje. Por supuesto, estos también son objetos (como verá, un objeto puede estar conformado por otros objetos).

¿Qué sucederá cuando invoque al caso de uso Lavar ropa? Si damos por hecho que completó las operaciones "agregar ropa", "agregar detergente" y "activar", la secuencia sería más o menos así:

1. El agua empezará a llenar el tambor mediante una manguera.
2. El tambor permanecerá inactivo durante cinco minutos.
3. La manguera dejará de abastecer agua.
4. El tambor girará de un lado a otro durante quince minutos.
5. El agua jabonosa saldrá por el drenaje.
6. Comenzará nuevamente el abastecimiento de agua.
7. El tambor continuará girando.

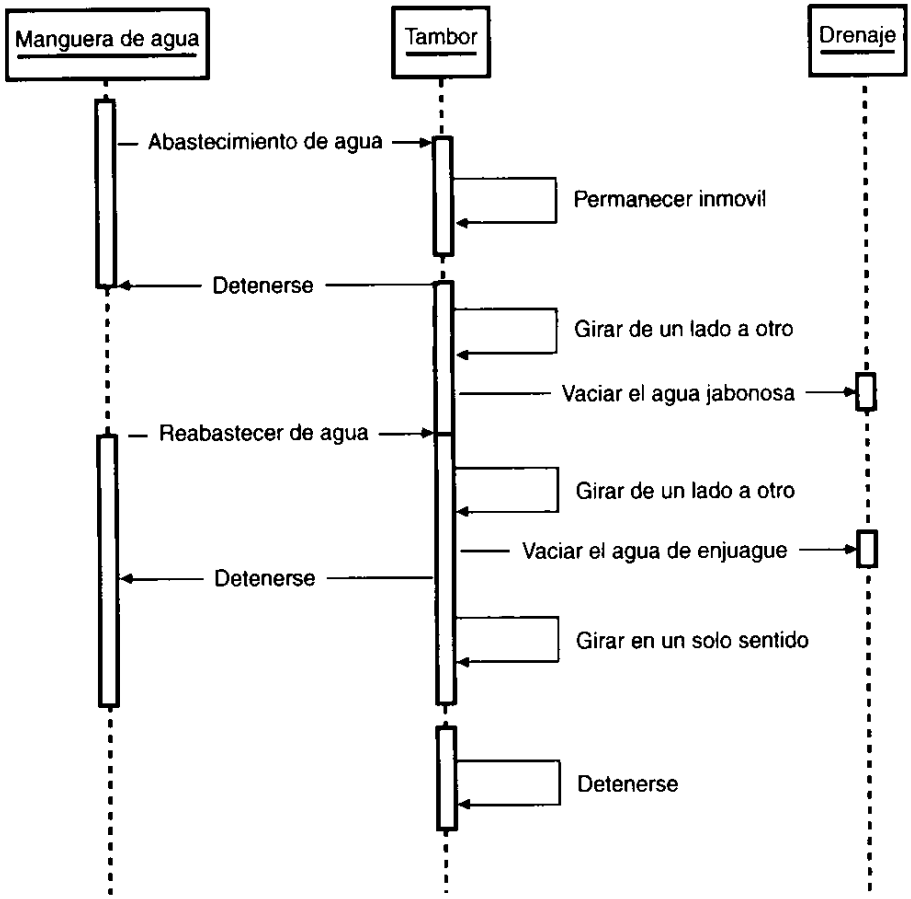
www.elsolucionario.net

www.elsolucionario.net

8. El abastecimiento de agua se detendrá.
9. El agua del enjuague saldrá por el drenaje.
10. El tambor girará en una sola dirección y se incrementará su velocidad por cinco minutos.
11. El tambor dejará de girar y el proceso de lavado habrá finalizado.

La figura 1.5 presenta un diagrama de secuencias que captura las interacciones que se realizan a través del tiempo entre el abastecimiento de agua, el tambor y el drenaje (representados como rectángulos en la parte superior del diagrama). En este diagrama el tiempo se da de arriba hacia abajo.

**FIGURA 1.5**
*Diagrama de secuencias UML.*




Por cierto, volviendo a las ideas acerca de los estados, podríamos caracterizar los pasos 1 y 2 como el estado de remojo, 3 y 4 como el estado de lavado, 5 a 7 como el estado de enjuague y del 8 al 10 como el estado de centrifugado.

## Diagrama de actividades

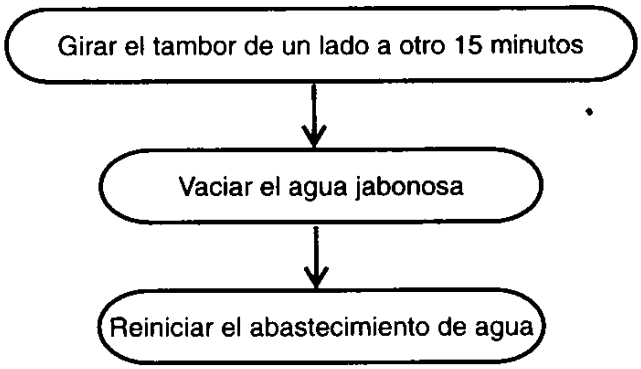
Las actividades que ocurren dentro de un caso de uso o dentro del comportamiento de un objeto se dan, normalmente, en secuencia, como en los once pasos de la sección anterior. La figura 1.6 muestra la forma en que el diagrama de actividades UML representa los pasos del 4 al 6 de tal secuencia.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 1.6**

*Diagrama de actividades UML.*

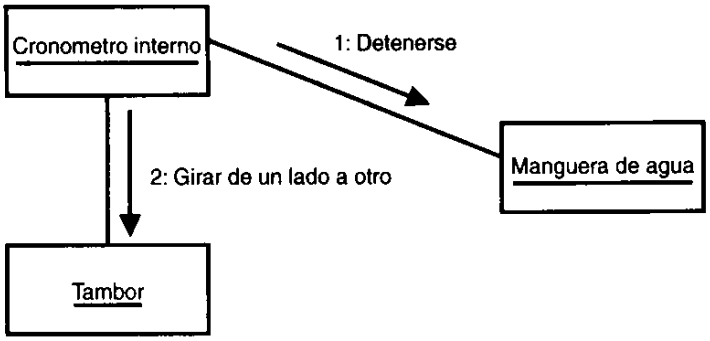



## Diagrama de colaboraciones

Los elementos de un sistema trabajan en conjunto para cumplir con los objetivos del sistema, y un lenguaje de modelado deberá contar con una forma de representar esto. El diagrama de colaboraciones UML, diseñado con este fin, se muestra en la figura 1.7. Este ejemplo agrega un cronómetro interno al conjunto de clases que constituyen a una lavadora. Luego de cierto tiempo, el cronómetro detendrá el flujo de agua y el tambor comenzará a girar de un lado a otro.

**FIGURA 1.7**

*Diagrama de colaboraciones UML.*



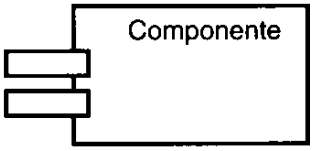

## Diagrama de componentes

Este diagrama y el siguiente dejarán el mundo de las lavadoras, dado que están íntimamente ligados con los sistemas informáticos.

El moderno desarrollo de software se realiza mediante componentes, lo que es particularmente importante en los procesos de desarrollo en equipo. Sin extenderme mucho en este punto le mostraré, en la figura 1.8, la manera en que el UML representa un componente de software.

**FIGURA 1.8**

*Diagrama de componentes UML.*




www.elsolucionario.net

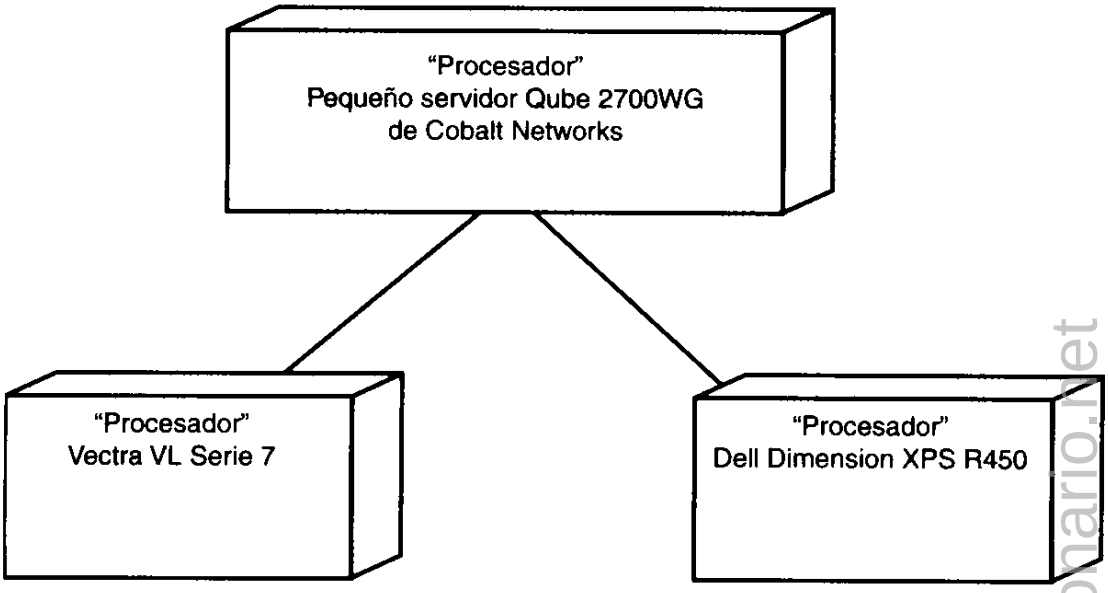
www.elsolucionario.net

## Diagrama de distribución

El diagrama de distribución UML muestra la arquitectura física de un sistema informático. Puede representar los equipos y dispositivos, mostrar sus interconexiones y el software que se encontrará en cada máquina. Cada computadora está representada por un cubo y las interacciones entre las computadoras están representadas por líneas que conectan a los cubos. La figura 1.9 presenta un ejemplo.

**FIGURA 1.9**

*Diagrama de distribución UML.*




# Otras características

Anteriormente, mencioné que el UML proporciona características que le permiten organizar y extender los diagramas.

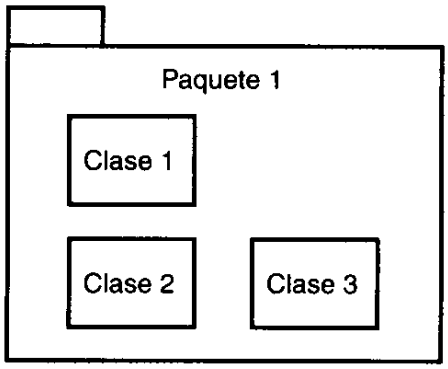
## Paquetes

TERMINO NUEVO

En algunas ocasiones se encontrará con la necesidad de organizar los elementos de un diagrama en un grupo. Tal vez quiera mostrar que ciertas clases o componentes son parte de un subsistema en particular. Para ello, los agrupará en un *paquete*, que se representará por una carpeta tabular, como se muestra en la figura 1.10.

**FIGURA 1.10**

*El paquete UML le permite agrupar los elementos de un diagrama.*




www.elsolucionario.net

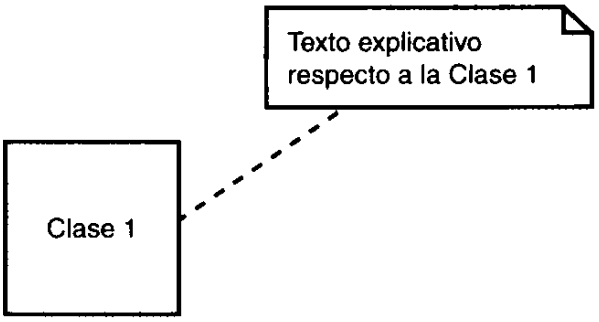
www.elsolucionario.net

## Notas

TÉRMINO NUEVO Es frecuente que alguna parte del diagrama no presente una clara explicación del porqué está allí o la manera en que trabaja. Cuando éste sea el caso, la *nota* UML será útil. Imagine a una nota como el equivalente gráfico de un papel adhesivo. La nota es un rectángulo con una esquina doblada, y dentro del rectángulo se coloca la explicación. Usted adjunta la nota al elemento del diagrama conectándolos mediante una línea discontinua.

**FIGURA 1.11**

*En cualquier diagrama, podrá agregar comentarios aclaratorios mediante una nota.*




## Estereotipos

TÉRMINO NUEVO El UML otorga varios elementos de utilidad, pero no es un conjunto minucioso de ellos. De vez en cuando diseñará un sistema que requiera algunos elementos hechos a la medida. Los *estereotipos* o clisés le permiten tomar elementos propios del UML y convertirlos en otros. Es como comprar un traje del mostrador y modificarlo para que se ajuste a sus medidas (contrario a confeccionarse uno completamente nuevo). Imagine a un estereotipo como este tipo de alteración. Lo representará como un nombre entre dos pares de paréntesis angulares y después los aplicará correctamente.

TÉRMINO NUEVO El concepto de una interfaz provee un buen ejemplo. Una interfaz es una clase que realiza operaciones y que no tiene atributos, es un conjunto de acciones que tal vez quiera utilizar una y otra vez en su modelo. En lugar de inventar un nuevo elemento para representar una interfaz, podrá utilizar el símbolo de una clase con «Interfaz» situada justo sobre el nombre de la clase, como se muestra en la figura 1.12.

**FIGURA 1.12**

*Un estereotipo le permite crear nuevos elementos a partir de otros existentes.*

www.elsolucionario.net

www.elsolucionario.net

# Para qué tantos diagramas

Como puede ver, los diagramas del UML le permiten examinar un sistema desde distintos puntos de vista. Es importante recalcar que en un modelo UML no es necesario que aparezcan todos los diagramas. De hecho, la mayoría de los modelos UML contienen un subconjunto de los diagramas que he indicado.

TERMINO NUEVO

¿Por qué es necesario contar con diferentes perspectivas de un sistema? Por lo general, un sistema cuenta con diversas personas implicadas las cuales tienen enfoques particulares en diversos aspectos del sistema. Volvamos al ejemplo de la lavadora. Si diseñara el motor de una lavadora, tendría una perspectiva del sistema; si escribiera las instrucciones de operación, tendría otra perspectiva. Si diseñara la forma general de la lavadora vería al sistema desde una perspectiva totalmente distinta a si tan sólo tratara de lavar su ropa.

El escrupuloso diseño de un sistema involucra todas las posibles perspectivas, y el diagrama UML le da una forma de incorporar una perspectiva en particular. El objetivo es satisfacer a cada persona implicada.

# Resumen

El desarrollo de sistemas es una actividad humana. Sin un sistema de notación fácil de comprender, el proceso de desarrollo tiene una gran cantidad de errores.

El UML es un sistema de notación que se ha convertido en estándar en el mundo del desarrollo de sistemas. Es el resultado del trabajo hecho por Grady Booch, James Rumbaugh e Ivar Jacobson. El UML está constituido por un conjunto de diagramas, y proporciona un estándar que permite al analista de sistemas generar un anteproyecto de varias facetas que sean comprensibles para los clientes, desarrolladores y todos aquellos que estén involucrados en el proceso de desarrollo. Es necesario contar con todos esos diagramas dado que cada uno se dirige a cada tipo de persona implicada en el sistema.

Un modelo UML indica *qué* es lo que supuestamente hará el sistema, mas no *cómo* lo hará.

# Preguntas y respuestas

**P He visto que se refiere al Lenguaje Unificado de Modelado como "UML" y como "el UML". ¿Cuál es el correcto?**

**R** Los creadores del lenguaje prefieren el uso de "el UML".

**P Ha indicado que el UML es adecuado para los analistas. No obstante, el diagrama de distribución no parece ser algo muy útil en la fase de análisis en el desarrollo de un sistema. ¿No sería más apropiado para una fase posterior?**

www.elsolucionario.net

www.elsolucionario.net

**R** En realidad nunca será demasiado pronto para empezar a pensar en la distribución (u otras cuestiones que, tradicionalmente, se dejan para fases posteriores del desarrollo). Aunque es cierto que el analista se interesa por hablar con los clientes y usuarios, en las fases tempranas del proceso el analista debería pensar en los equipos y componentes que constituirían el hardware del sistema. En algunas ocasiones, el cliente dicta esto; en otras, el cliente desea una recomendación del equipo de desarrollo. Ciertamente, un arquitecto de sistemas encontrará útil al diagrama de distribución.

**P Ha mencionado que es posible hacer diagramas híbridos. ¿UML, perdón, *el* UML, impone limitaciones respecto a los elementos que podrá combinar en un diagrama?**

**R** No. El UML no establece límites, no obstante, con frecuencia se da el caso de que un diagrama contenga un tipo de elemento. Podrá colocar símbolos de clases en un diagrama de distribución, pero ello no será muy útil.

# Taller

Ya se ha iniciado en el UML. Ahora deberá reafirmar su conocimiento de esta gran herramienta al responder algunas preguntas y realizar los ejercicios. Las respuestas aparecerán en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿Porqué es necesario contar con diversos diagramas en el modelo de un sistema?
2. ¿Cuáles diagramas le dan una perspectiva estática de un sistema?
3. ¿Cuáles diagramas le dan una perspectiva dinámica de un sistema (esto es, muestran el cambio progresivo)?

## Ejercicios

1. Suponga que creará un sistema informático que jugará ajedrez con un usuario. ¿Cuáles diagramas UML serían útiles para diseñar el sistema? ¿Por qué?
2. Para el sistema del ejercicio que ha completado, liste las preguntas que formularía a un usuario potencial y por qué las haría.

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 2

# Orientación a objetos

Es fundamental que comprenda todo lo relacionado a la orientación a objetos para el proceso que realizará; específicamente, es importante que conozca algunos conceptos sobre la orientación a objetos.

En esta hora se tratarán los siguientes temas:

- Abstracción
- Herencia
- Polimorfismo
- Encapsulamiento o encapsulación
- Envío de mensajes
- Asociaciones
- Agregación

La orientación a objetos ha tomado por asalto y en forma legítima al mundo del software. Como medio para la generación de programas, tiene varias ventajas. Fomenta una metodología basada en componentes para el desarrollo

www.elsolucionario.net

www.elsolucionario.net

de software, de manera que primero se genera un sistema mediante un conjunto de objetos, luego podrá ampliar el sistema agregándole funcionalidad a los componentes que ya había generado o agregándole nuevos componentes, y finalmente podrá volver a utilizar los objetos que generó para el sistema cuando cree uno nuevo, con lo cual reducirá sustancialmente el tiempo de desarrollo de un sistema.

La orientación a objetos es tan importante para el diseño de software que el OMG (Grupo de administración de objetos), una corporación no lucrativa que establece las normas para el desarrollo orientado a objetos, predice que los ingresos obtenidos por el software orientado a objetos serán de 3 millardos de dólares en los siguientes tres a cinco años. El UML influye en esto al permitirle generar modelos de objetos fáciles de usar y comprender para que los desarrolladores puedan convertirlos en software.

La orientación a objetos es un paradigma (un paradigma que depende de ciertos principios fundamentales). En esta hora comprenderá dichos principios y verá qué es lo que hace funcionar a los objetos y cómo utilizarlos en el análisis y diseño. En la siguiente hora, empezará a aplicar el UML a tales principios.

# Objetos, objetos por doquier

Los objetos concretos y virtuales, están a nuestro alrededor, ellos conforman nuestro mundo. Como indiqué en la hora anterior, el software actual simula al mundo (o un segmento de él), y los programas, por lo general, imitan a los objetos del mundo. Si comprende algunas cuestiones básicas de los objetos, entenderá cómo se deben mostrar éstos en las representaciones de software.

**TERMINO NUEVO** Antes que nada, un objeto es la instancia de una clase (o categoría). Usted y yo, por ejemplo, somos instancias de la clase Persona. Un objeto cuenta con una *estructura*, es decir atributos (propiedades) y acciones. Las acciones son todas las *actividades* que el objeto es capaz de realizar. Los atributos y acciones, en conjunto, se conocen como *características* o *rasgos*.

Como objetos de la clase Persona, usted y yo contamos con los siguientes atributos: altura, peso y edad (puede imaginar muchos más). También realizamos las siguientes tareas: comer, dormir, leer, escribir, hablar, trabajar, etcétera. Si tuviéramos que crear un sistema que manejara información acerca de las personas (como una nómina o un sistema para el departamento de recursos humanos), sería muy probable que incorporáramos algunos de sus atributos y acciones en nuestro software.

En el mundo de la orientación a objetos, una clase tiene otro propósito además de la categorización. En realidad es una plantilla para fabricar objetos. Imagínelo como un molde de galletas que produce muchas galletas (algunos alegarían que esto es lo mismo que la categorización, pero evitemos dicho debate).

www.elsolucionario.net

www.elsolucionario.net

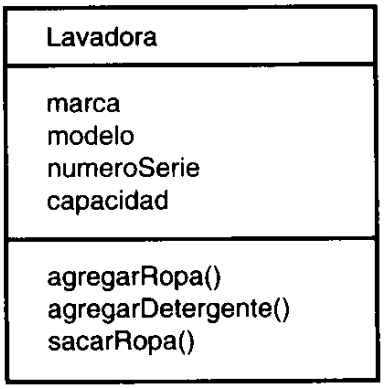
Regresemos al ejemplo de la lavadora. Si en la clase Lavadora se indica la marca, el modelo, el número de serie y la capacidad (junto con las acciones de agregar ropa, agregar detergente y sacar ropa), tendrá un mecanismo para fabricar nuevas instancias a partir de su clase; es decir, podrá crear nuevos objetos (vea la figura 2.1).

En la hora 3, "Uso de la orientación a objetos", verá que los nombres de las clases, como *lavadora*, se escribirán como Lavadora, y si constara de dos palabras se escribiría como LavadoraIndustrial, y las características como número *de serie* se escribirán como numeroSerie.

Esto es particularmente importante en el desarrollo de software orientado a objetos. Aunque este libro no se enfoca a la programación, le ayudará a comprender la orientación a objetos si sabe que las clases en los programas orientados a objetos pueden crear nuevas instancias.

**FIGURA 2.1**

*La clase Lavadora (modelo original) es una plantilla para generar nuevas instancias de Lavadoras.*



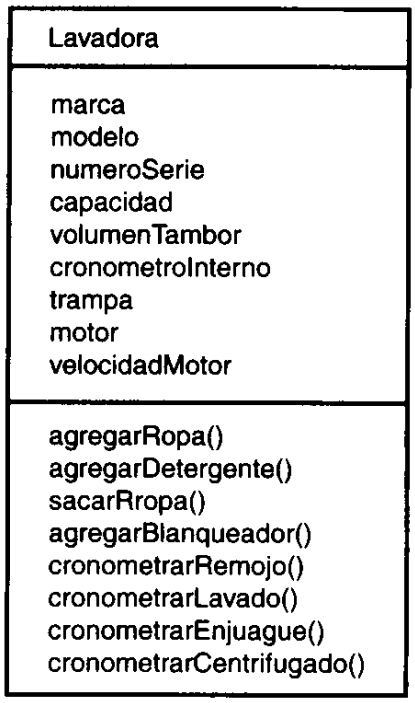

Es importante que recuerde que el propósito de la orientación a objetos es desarrollar software que refleje particularmente (es decir, que modele) un esquema del mundo. Entre más atributos y acciones tome en cuenta, mayor será la similitud de su modelo con la realidad. En el ejemplo de la lavadora, tendrá un modelo más exacto si incluye los siguientes atributos: volumen del tambor, cronómetro interno, trampa, motor y velocidad del motor. Podría hacerlo más preciso si incluye las acciones de agregar blanqueador, cronometrar el remojo, cronometrar el lavado, cronometrar el enjuague y cronometrar el centrifugado (vea la figura 2.2).

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 2.2**

*La adición de atributos y acciones al modelo lo acerca a la realidad.*




# Algunos conceptos

La orientación a objetos se refiere a algo más que tan sólo atributos y acciones; también considera otros aspectos. Dichos aspectos se conocen como *abstracción*, *herencia*, *polimorfismo* y *encapsulamiento* o *encapsulación*. Otros aspectos importantes de la orientación a objetos son: el *envío de mensajes*, las *asociaciones* y la *agregación*. Examinemos cada uno de estos conceptos.

## Abstracción

TERMINO NUEVO

La abstracción se refiere a quitar las propiedades y acciones de un objeto para dejar sólo aquellas que sean necesarias. ¿Qué significa esto último?

Diferentes tipos de problemas requieren distintas cantidades de información, aun si estos problemas pertenecen a un área en común. En la segunda fase de la creación de la clase Lavadora, se podrían agregar más atributos y acciones que en la primera fase. ¿Vale la pena?

Valdría la pena si usted pertenece al equipo de desarrollo que generará finalmente la aplicación que simule con exactitud lo que hace una lavadora. Un programa de este tipo (que podría ser muy útil para los ingenieros de diseño que actualmente estén trabajando en el diseño de una lavadora) deberá ser tan completo que permita obtener predicciones exactas respecto a lo que ocurriría cuando se fabrique la lavadora, funcione a toda su capacidad y lave la ropa. De hecho, para este caso podrá quitar el atributo del número de serie, dado que posiblemente no será de mucha ayuda.

Por otra parte, si va a generar un software que haga un seguimiento de las transacciones en una lavandería que cuente con diversas lavadoras, posiblemente no valdrá la pena. En este programa no necesitará todos los atributos detallados y operaciones del párrafo anterior, no obstante, quizá necesite incluir el número de serie de cada objeto Lavadora.

www.elsolucionario.net

www.elsolucionario.net

En cualquier caso, con lo que se quedará luego de tomar su decisión respecto a lo que incluirá o desechará, será una abstracción de una lavadora.

## Herencia

TERMINO NUEVO

Como ya se mencionó anteriormente, una clase es una categoría de objetos (y en el mundo del software, una plantilla sirve para crear otros objetos). Un objeto es una instancia de una clase. Esta idea tiene una consecuencia importante: como instancia de una clase, un objeto tiene todas las características de la clase de la que proviene. A esto se le conoce como *herencia*. No importa qué atributos y acciones decida usar de la clase Lavadora, cada objeto de la clase heredará dichos atributos y operaciones.

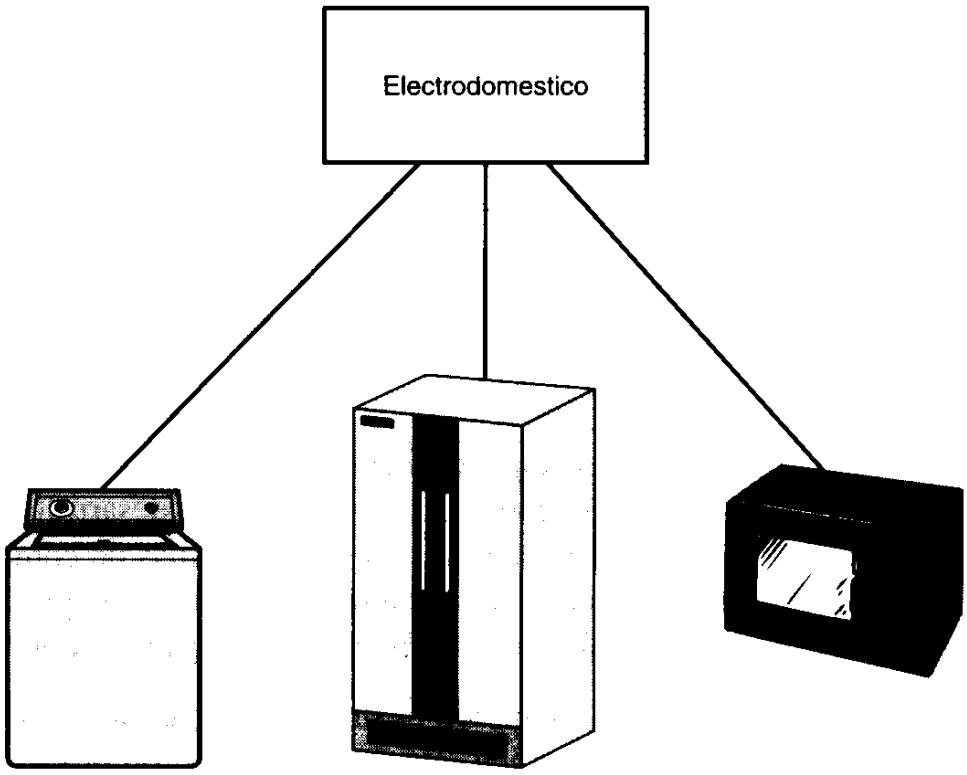
Un objeto no sólo hereda de una clase, sino que una clase también puede heredar de otra. Las lavadoras, refrigeradores, hornos de microondas, tostadores, lavaplatos, radios, licuadoras y planchas son clases y forman parte de una clase más genérica llamada: Electrodomesticos. Un electrodoméstico cuenta con los atributos de interruptor y cable eléctrico, y las operaciones de encendido y apagado. Cada una de las clases Electrodomestico heredará los mismos atributos; por ello, si sabe que algo es un electrodoméstico, de inmediato sabrá que cuenta con los atributos y acciones de la clase Electrodomestico.

TERMINO NUEVO

Otra forma de explicarlo es que la lavadora, refrigerador, horno de microondas y cosas por el estilo son *subclases* de la clase Electrodomestico. Podemos decir que la clase Electrodomestico es una *superclase* de todas las demás. La figura 2.3 le muestra la relación de superclase y subclase.

**FIGURA 2.3**

*Los electrodomésticos heredan los atributos y acciones de la clase Electrodomestico. Cada electrodoméstico es una subclase de la clase Electrodomestico. La clase Electrodomestico es una superclase de cada subclase.*



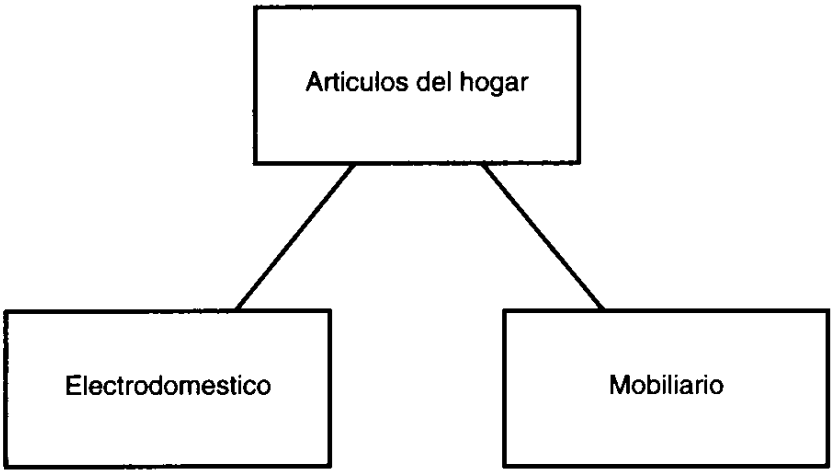

www.elsolucionario.net

www.elsolucionario.net

La herencia no tiene por qué terminar aquí. Por ejemplo, Electrodomestico es una subclase de Articulos del hogar, como le muestra la figura 2.4. Otra de las subclases de Articulos del hogar podría ser Mobiliario, que tendrá sus propias subclases.

**FIGURA 2.4**

*Las superclases también pueden ser subclases, y heredar de otras superclases.*



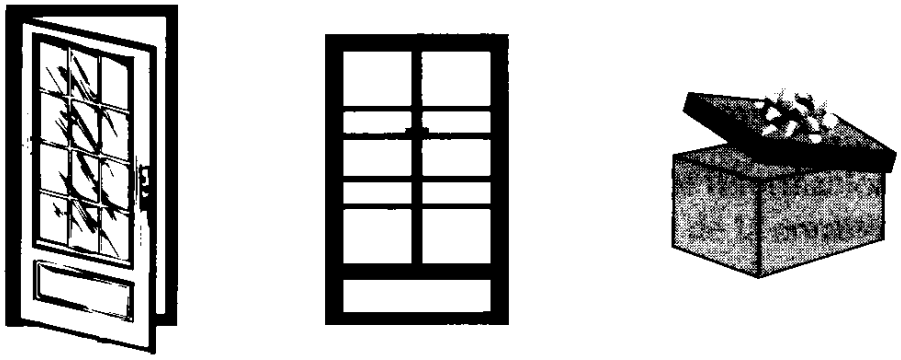

## Polimorfismo

TERMINO NUEVO

En ocasiones una operación tiene el mismo nombre en diferentes clases. Por ejemplo, podrá abrir una puerta, una ventana, un periódico, un regalo o una cuenta de banco, en cada uno de estos casos, realizará una operación diferente. En la orientación a objetos, cada clase "sabe" cómo realizar tal operación. Esto es el *polimorfismo* (vea la figura 2.5).

**FIGURA 2.5**

*En el polimorfismo, una operación puede tener el mismo nombre en diversas clases, y funcionar distinto en cada una.*



En primera instancia, parecería que este concepto es más importante para los desarrolladores de software que para los modeladores, ya que los desarrolladores tienen que crear el software que implemente tales métodos en los programas de computación, y deben estar conscientes de diferencias importantes entre las operaciones que pudieran tener el mismo nombre. Y podrán generar clases de software que "sepan" lo que se supone que harán.

No obstante, el polimorfismo también es importante para los modeladores ya que les permite hablar con el cliente (quien está familiarizado con la sección del mundo que será modelada) en las propias palabras y terminología del cliente. En ocasiones, las palabras y terminología del cliente nos conducen a palabras de acción (como "abrir") que pueden

www.elsolucionario.net

www.elsolucionario.net

tener más de un significado. El polimorfismo permite al modelador mantener tal terminología sin tener que crear palabras artificiales para sustentar una unicidad innecesaria de los términos.

## Encapsulamiento

En cierto comercial televisivo, dos personas discuten acerca de todo el dinero que ahorrarían si marcaran un prefijo telefónico en particular antes de hacer una llamada de larga distancia.

Uno de ellos pregunta con incredulidad: "¿Cómo funciona?"

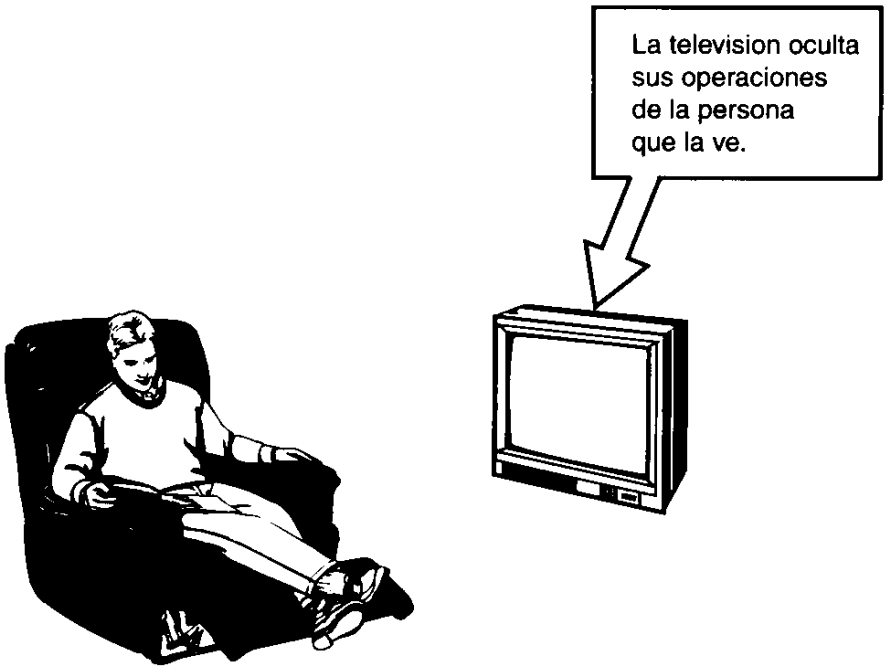
Y el otro responde: "¿Cómo hacen que las rosetas de maíz estallen? ¿A quién le importa?"

**Término Nuevo**

La esencia del *encapsulamiento* (o encapsulación) es que cuando un objeto trae consigo su funcionalidad, esta última se oculta (vea la figura 2.6). Por lo general, la mayoría de la gente que ve la televisión no sabe o no se preocupa de la complejidad electrónica que hay detrás de la pantalla ni de todas las operaciones que tienen que ocurrir para mostrar una imagen en la pantalla. La televisión hace lo que tiene que hacer sin mostrarnos el proceso necesario para ello y, por suerte, la mayoría de los electrodomésticos funcionan así.

**Figura 2.6**

*Los objetos encapsulan lo que hacen; es decir, ocultan la funcionalidad interna de sus operaciones, de otros objetos y del mundo exterior.*




¿Cuál es la importancia de esto? En el mundo del software, el encapsulamiento permite reducir el potencial de errores que pudieran ocurrir. En un sistema que consta de objetos, éstos dependen unos de otros en diversas formas. Si uno de ellos falla y los especialistas de software tienen que modificarlo de alguna forma, el ocultar sus operaciones de otros objetos significará que tal vez no será necesario modificar los demás objetos.

www.elsolucionario.net

www.elsolucionario.net

En el mundo real, también verá la importancia del encapsulamiento en los objetos con los que trabaje. Por ejemplo, el monitor de su computadora, en cierto sentido, oculta sus operaciones de la CPU, es decir, si algo falla en su monitor, lo reparará o lo reemplazará; pero es muy probable que no tenga que reparar o reemplazar la CPU al mismo tiempo que el monitor.

TERMINO NUEVO

Ya que estamos en el tema, existe un concepto relacionado. Un objeto oculta lo que hace a otros objetos y al mundo exterior, por lo cual al encapsulamiento también se le conoce como *ocultamiento de la información*. Pero un objeto tiene que presentar un "rostro" al mundo exterior para poder iniciar sus operaciones. Por ejemplo, la televisión tiene diversos botones y perillas en sí misma o en el control remoto. Una lavadora tiene diversas perillas que le permiten establecer los niveles de temperatura y agua. Los botones y perillas de la televisión y de la lavadora se conocen como *interfaces*.

## Envío de mensajes

Mencioné que en un sistema los objetos trabajan en conjunto. Esto se logra mediante el envío de mensajes entre ellos. Un objeto envía a otro un mensaje para realizar una operación, y el objeto receptor ejecutará la operación.

Una televisión y su control remoto pueden ser un ejemplo muy intuitivo del mundo que nos rodea. Cuando desea ver un programa de televisión, busca el control remoto, se sienta en su silla favorita y presiona el botón de encendido. ¿Qué ocurre? El control remoto le envía, literalmente, un mensaje al televisor para que se encienda. El televisor recibe el mensaje, lo identifica como una petición para encenderse y así lo hace. Cuando desea ver otro canal, presiona el botón correspondiente del control remoto, mismo que envía otro mensaje a la televisión (cambiar de canal). El control remoto también puede comunicar otros mensajes como ajustar el volumen, silenciar y activar los subtítulos.

Volvamos a las interfaces. Muchas de las cosas que hace mediante el control remoto, también las podrá hacer si se levanta de la silla, va a la televisión y presiona los botones correspondientes (¡alguna vez lo habrá hecho ya!). La interfaz que la televisión le presenta (el conjunto de botones y perillas) no es, obviamente, la misma que le muestra al control remoto (un receptor de rayos infrarrojos). La figura 2.7 le muestra esto.

## Asociaciones

Otro acontecimiento común es que los objetos se relacionan entre sí de alguna forma. Por ejemplo, cuando enciende su televisor, en términos de orientación a objetos, usted se asocia con su televisor.

La asociación "encendido" es en una sola dirección (una vía), esto es, usted enciende la televisión, como se ve en la figura 2.8. No obstante, a menos que vea demasiada televisión, ella no le devolverá el favor. Hay otras asociaciones que son en dos direcciones, como "casamiento".

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 2.7**

*Ejemplo de un mensaje enviado de un objeto a otro: el objeto "control remoto" envía un mensaje al objeto "televisión" para encenderse. El objeto "televisión" recibe el mensaje mediante su interfaz, un receptor infrarrojo.*

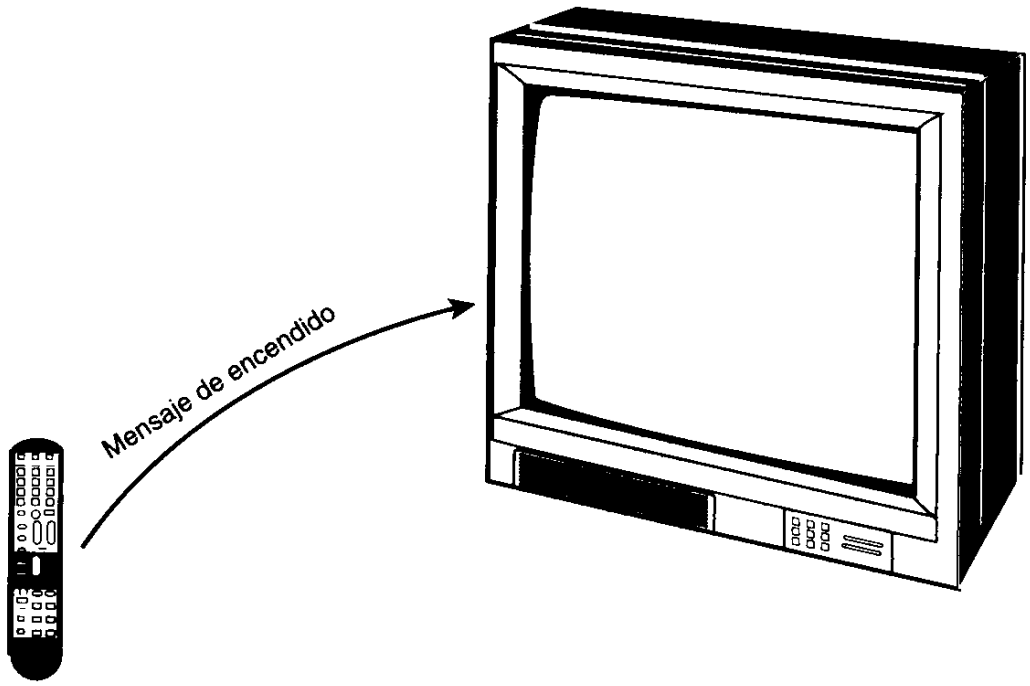
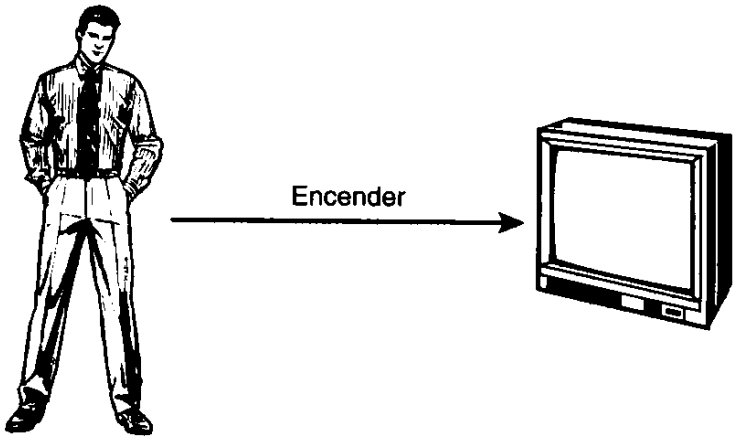



**FIGURA 2.8**

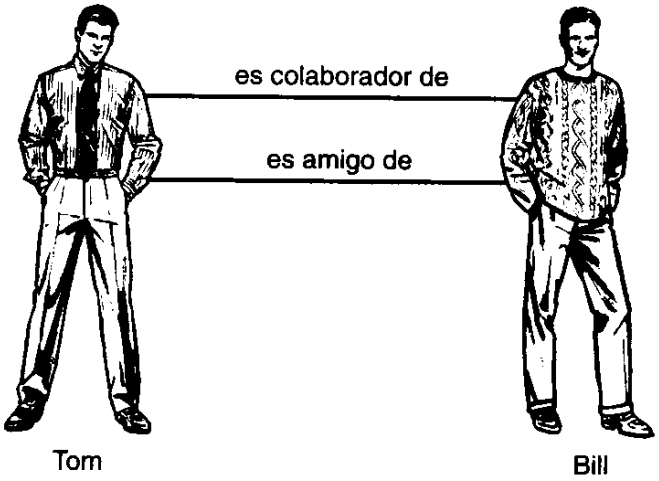
*Con frecuencia los objetos se relacionan entre sí de alguna forma. Cuando usted enciende su televisión, tendrá una asociación en una sola dirección con ella.*




En ocasiones, un objeto podría asociarse con otro en más de una forma. Si usted y su colaborador son amigos, ello servirá de ejemplo. Usted tendría una asociación "es amigo de", así como "es colaborador de", como se aprecia en la figura 2.9.

**FIGURA 2.9**

*En ocasiones, los objetos pueden asociarse en más de una forma.*



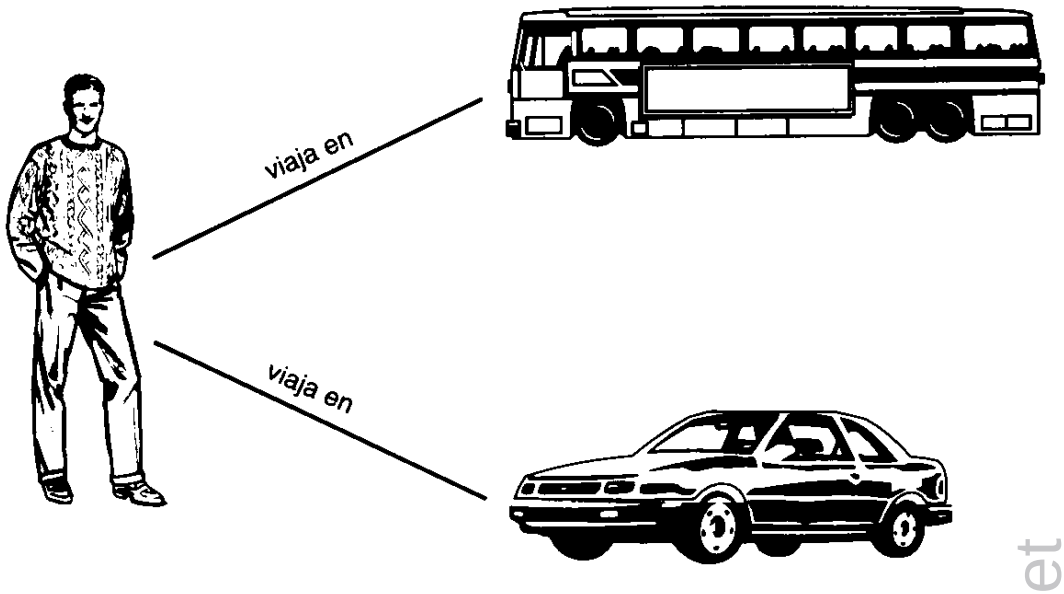

www.elsolucionario.net

www.elsolucionario.net

Una clase se puede asociar con más de una clase distinta. Una persona puede viajar en automóvil, pero también puede hacerlo en autobús (vea la figura 2.10).

**FIGURA 2.10**

*Una clase puede asociarse con más de una clase distinta.*




TÉRMINO NUEVO La *multiplicidad* (o diversificación) es un importante aspecto de las asociaciones entre objetos. Indica la cantidad de objetos de una clase que se relacionan con otro objeto en particular de la clase asociada. Por ejemplo, en un curso escolar, el curso se imparte por un solo instructor, en consecuencia, el curso y el instructor están en una asociación de uno a uno. Sin embargo, en un seminario hay diversos instructores que impartirán el curso a lo largo del semestre, por lo tanto, el curso y el instructor tienen una asociación de uno a muchos.

Podrá encontrar todo tipo de multiplicidades si echa una mirada a su alrededor. Una bicicleta rueda en dos neumáticos (multiplicidad de uno a dos), un triciclo rueda en tres, y un vehículo de 18 ruedas, en 18.

## Agregación

Vea su computadora: cuenta con un gabinete, un teclado, un ratón, un monitor, una unidad de CD-ROM, uno o varios discos duros, un módem, una unidad de disquete, una impresora y, posiblemente, bocinas. Dentro del gabinete, junto con las citadas unidades de disco, tiene una CPU, una tarjeta de vídeo, una de sonido y otros elementos sin los que, sin duda, difícilmente podría vivir.

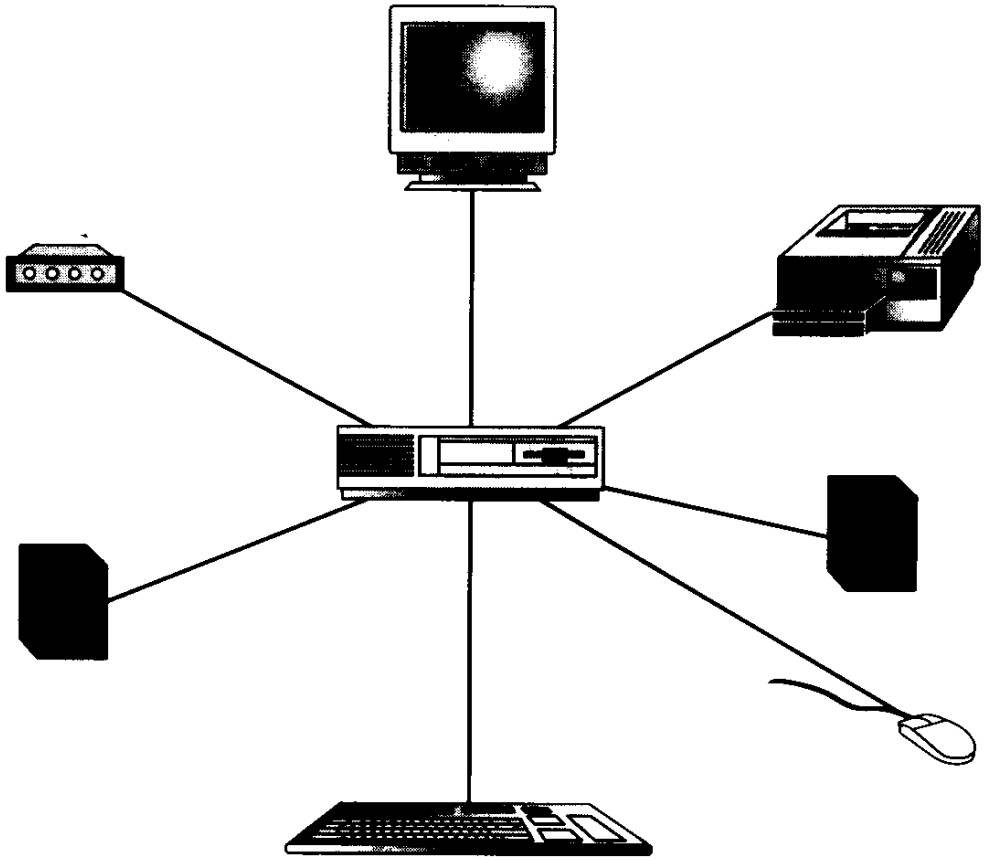
TÉRMINO NUEVO Su computadora es una *agregación* o adición, otro tipo de asociación entre objetos. Como muchas otras cosas que valdrían la pena tener, su equipo está constituido de diversos tipos de componentes (vea la figura 2.11). Tal vez conozca varios ejemplos de agregaciones.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 2.11**

*Una computadora es un ejemplo de agregación: un objeto que se conforma de una combinación de diversos tipos de objetos.*



**TÉRMINO NUEVO** Un tipo de agregación trae consigo una estrecha relación entre un objeto agregado y sus objetos componentes. A esto se le conoce como *composición*. El punto central de la composición es que el componente se considera como tal sólo como parte del objeto compuesto. Por ejemplo: una camisa está compuesta de cuerpo, cuello, mangas, botones, ojales y puños. Suprima la camisa y el cuello será inútil.

En ocasiones, un objeto compuesto no tiene el mismo tiempo de vida que sus propios componentes. Las hojas de un árbol pueden morir antes que el árbol. Si destruye al árbol, también las hojas morirán (vea la figura 2.12).

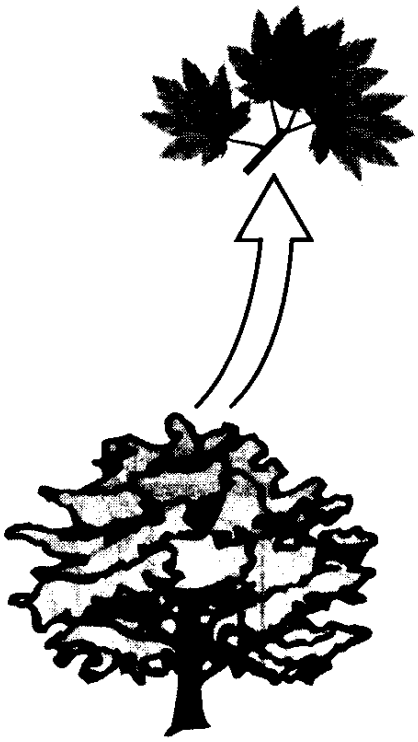
La agregación y la composición son importantes dado que reflejan casos extremadamente comunes, y ello ayuda a que cree modelos que se asemejen considerablemente a la realidad.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 2.12**

*En una composición, un componente puede morir antes que el objeto compuesto. Si destruye al objeto compuesto, destruirá también a los componentes.*



# La recompensa

Los objetos y sus asociaciones conforman la columna vertebral de la funcionalidad de los sistemas. Para modelarlos, necesitará comprender lo que son las asociaciones. Si está consciente de los posibles tipos de asociaciones, tendrá una amplia gama de posibilidades cuando hable con los clientes respecto a sus necesidades, obtendrá sus requerimientos y creará los modelos de los sistemas que los ayudarán a cumplir con sus retos de negocios.

**TÉRMINO NUEVO** Lo importante es utilizar los conceptos de la orientación a objetos para ayudarle a comprender el área de conocimiento de su cliente (su *dominio*), y esclarecer sus puntos de vista al cliente en términos que él o ella puedan comprender.

Es allí donde se aplica el UML. En las siguientes tres horas, aprenderá a aplicar el UML para visualizar los conceptos que ha aprendido durante esta hora.

# Resumen

La orientación a objetos es un paradigma que depende de algunos principios fundamentales. Un objeto es una instancia de una clase. Una clase es una categoría genérica de objetos que tienen los mismos atributos y acciones. Cuando crea un objeto, el área del problema en que trabaje determinará cuántos de los atributos y acciones debe tomar en cuenta.

La herencia es un aspecto importante de la orientación a objetos, un objeto hereda los atributos y operaciones de su clase. Una clase también puede heredar atributos y acciones de otra.

www.elsolucionario.net

www.elsolucionario.net

El polimorfismo es otro aspecto importante, ya que especifica que una acción puede tener el mismo nombre en diferentes clases y cada clase ejecutará tal operación de forma distinta.

Los objetos ocultan su funcionalidad de otros objetos y del mundo exterior. Cada objeto presenta una interfaz para que otros objetos (y personas) puedan aprovechar su funcionalidad.

Los objetos funcionan en conjunto mediante el envío de mensajes entre ellos. Los mensajes son peticiones para realizar operaciones.

Por lo general, los objetos se asocian entre sí y esta asociación puede ser de diversos tipos. Un objeto en una clase puede asociarse con cualquier cantidad de objetos distintos en otra clase.

La agregación es un tipo de asociación. Un objeto agregado consta de un conjunto de objetos que lo componen y una composición es un tipo especial de agregación. En un objeto compuesto, los componentes sólo existen como parte del objeto compuesto.

# Preguntas y respuestas

**P Usted dijo que la orientación a objetos ha tomado por asalto al mundo del software. ¿Qué no hay algunas aplicaciones importantes que no están orientadas a objetos?**

**R** Sí, y se conocen como sistemas "heredados" (programas que en muchos casos son ejecutados para mostrar su época). La orientación a objetos ofrece diversas ventajas, como la reusabilidad y un rápido periodo de desarrollo. Por tales razones, muy probablemente verá las nuevas aplicaciones (y las versiones rediseñadas de varias aplicaciones antiguas) escritas bajo el esquema de la orientación a objetos.

www.elsolucionario.net

# Taller

Para repasar lo que ha aprendido de la orientación a objetos, intente responder a algunas preguntas y realizar los siguientes ejercicios. Las respuestas las encontrará en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿Qué es un objeto?
2. ¿Cómo trabajan los objetos en conjunto?
3. ¿Qué establece la multiplicidad?
4. ¿Pueden asociarse dos objetos entre sí en más de una manera?

www.elsolucionario.net

## Ejercicios

Esta es una hora teórica, así que no incluí ejercicios. Verá algunos en las siguientes horas.

www.elsolucionario.net

www.elsolucionario.net

# HORA 3

# Uso de la orientación a objetos

A continuación conjugaremos las características del UML con los conceptos de la orientación a objetos. Aquí reafirmará su conocimiento de la orientación a objetos al tiempo que aprenderá otras cosas del UML.

En esta hora se tratarán los siguientes temas:

- Concepción de una clase
- Atributos
- Operaciones
- Responsabilidades y restricciones
- Qué es lo que hacen las clases y cómo encontrarlas

## Concepción de una clase

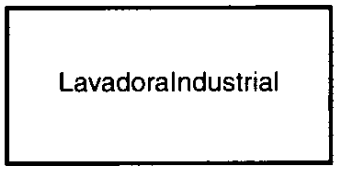
Como lo indiqué en la primera hora, en el UML un rectángulo es el símbolo que representa una clase. El nombre de la clase es, por convención, una palabra con la primera letra en mayúscula y normalmente se coloca en la parte superior del rectángulo. Si el nombre de su clase consta de dos palabras, únalas e inicie cada una con mayúscula (como en LavadoraIndustrial en la figura 3.1).

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 3.1**

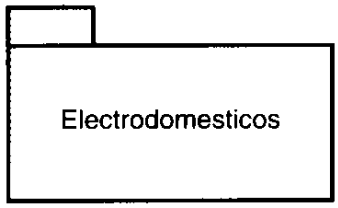
*La representación UML de una clase.*




Otra estructura del UML, el paquete, puede jugar un papel en el nombre de la clase. Como indiqué en la hora 1, "Introducción al UML", un paquete es la manera en que el UML organiza un diagrama de elementos. Tal vez recuerde que el UML representa un paquete como una carpeta tabular cuyo nombre es una cadena de texto (vea la figura 3.2).

**FIGURA 3.2**

*Un paquete del UML.*

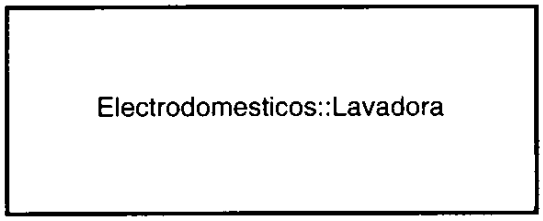



**TÉRMINO NUEVO** Si la clase "Lavadora" es parte de un paquete llamado "Electrodomesticos", podrá darle el nombre "Electrodomesticos::Lavadora". El par de dos puntos separa al nombre del paquete, que está a la izquierda, del nombre de la clase, que va a la derecha. A este tipo de nombre de clase se le conoce como *nombre de ruta* (vea la figura 3.3).

Posiblemente haya notado que en los nombres se han evitado los caracteres acentuados (como en Electrodomesticos) y la letra eñe. Esto se debe a que en el alfabeto inglés, tales caracteres no están contemplados y no podemos asegurar que el utilizarlos en sus identificadores no le traiga problemas, tanto en el UML como en el lenguaje de programación que piense utilizar para traducir los modelos. Por ello, evitaremos los acentos en todos los diagramas que se presentan a lo largo de este libro, de igual manera, evitaremos el uso de la letra eñe, misma que sustituiremos —en su caso— por "ni" (como en Anio, en lugar de Año).

**FIGURA 3.3**

*Una clase con un nombre de ruta.*




# Atributos

**TÉRMINO NUEVO** Un *atributo* es una propiedad o característica de una clase y describe un rango de valores que la propiedad podrá contener en los objetos (esto es, instancias) de la clase. Una clase podrá contener varios o ningún atributo. Por convención, si el atributo consta de una sola palabra se escribe en minúsculas; por otro lado, si el nombre contiene más de una palabra, cada palabra será unida a la anterior y comenzará con una letra mayúscula, a excepción de la primer palabra que comenzará en minúscula. La lista

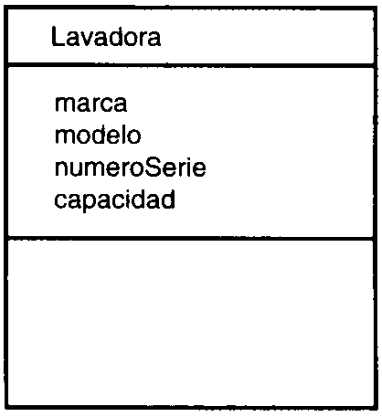
www.elsolucionario.net

www.elsolucionario.net

de nombres de atributos iniciará luego de una línea que la separe del nombre de la clase, como se aprecia en la figura 3.4.

**FIGURA 3.4**

*Una clase y sus atributos.*

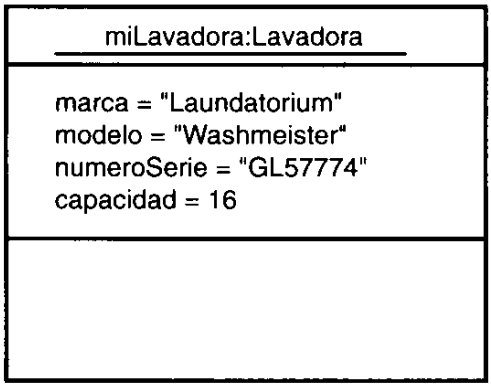



Todo objeto de la clase tiene un valor específico en cada atributo. La figura 3.5 le muestra un ejemplo. Observe que el nombre de un objeto inicia con una letra minúscula, y está precedido de dos puntos que a su vez están precedidos del nombre de la clase, y todo el nombre está subrayado.

El nombre `miLavadora:Lavadora` es una *instancia con nombre*; pero también es posible tener una *instancia anónima*, como `:Lavadora`.

**FIGURA 3.5**

*Un objeto cuenta con un valor específico en cada uno de los atributos que lo componen.*




El UML le da la opción de indicar información adicional de los atributos. En el símbolo de la clase, podrá especificar un tipo para cada valor del atributo. Entre los posibles tipos se encuentran cadena (string), número de punto flotante (float), entero (integer) y booleano (boolean), así como otros tipos enumerados. Para indicar un tipo, utilice dos puntos (:) para separar el nombre del atributo de su tipo. También podrá indicar un valor predeterminado para un atributo. La figura 3.6 le muestra las formas de establecer atributos.

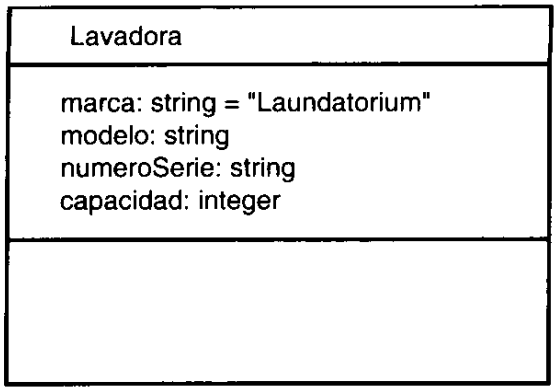
Aunque no parece haber restricción en la designación de tipos a las variables, utilizaremos los nombres en inglés para ceñirnos a los tipos que aparecen en los lenguajes de programación.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 3.6**

*Un atributo puede mostrar su tipo así como su valor predeterminado.*

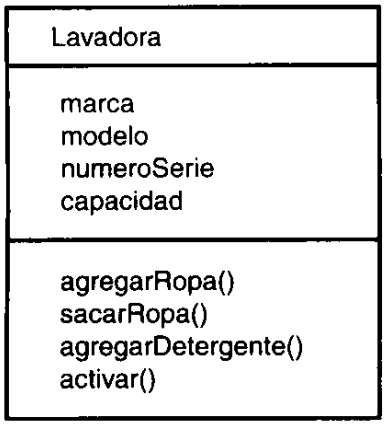



# Operaciones

**TÉRMINO NUEVO** Una *operación* es algo que la clase puede realizar, o que usted (u otra clase) pueden hacer a una clase. De la misma manera que el nombre de un atributo, el nombre de una operación se escribe en minúsculas si consta de una sola palabra. Si el nombre constara de más de una palabra, únalas e inicie todas con mayúscula exceptuando la primera. La lista de operaciones se inicia debajo de una línea que separa a las operaciones de los atributos, como se muestra en la figura 3.7.

**FIGURA 3.7**

*La lista de operaciones de una clase aparece debajo de una línea que las separa de los atributos de la clase.*




**TÉRMINO NUEVO** Así como es posible establecer información adicional de los atributos, también lo es en lo concerniente a las operaciones. En los paréntesis que preceden al nombre de la operación podrá mostrar el parámetro con el que funcionará la operación junto con su tipo de dato. La *función*, que es un tipo de operación, devuelve un valor luego que finaliza su trabajo. En una función podrá mostrar el tipo de valor que regresará.

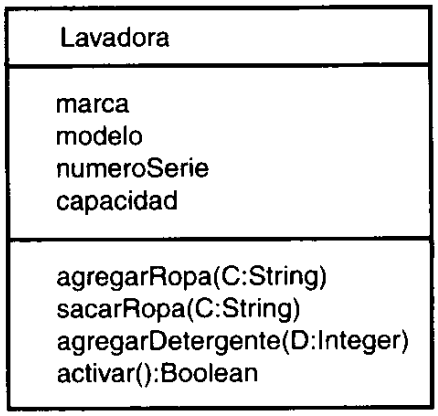
**TÉRMINO NUEVO** Estas secciones de información acerca de una operación se conocen como la *firma* de la operación. La figura 3.8 le muestra cómo representar la firma.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 3.8**

*La firma de una operación.*

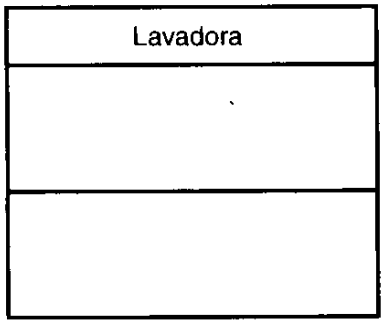



# Atributos, operaciones y concepción

Hasta ahora, hemos tratado a las clases como entidades aisladas, y hemos visto todos los atributos y operaciones de una clase. No obstante, en la práctica mostrará más de una clase a la vez; cuando lo haga, no será muy útil que siempre aparezcan todos los atributos y operaciones, ya que el hacerlo le crearía un diagrama muy saturado. En lugar de ello podrá tan sólo mostrar el nombre de la clase y dejar ya sea el área de atributos o el de operaciones (o ambas) vacía, como se muestra en la figura 3.9.

**FIGURA 3.9**

*En la práctica, no siempre mostrará todos los atributos y operaciones de una clase.*

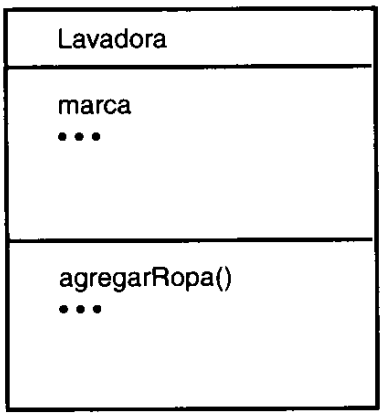



**TERMINO NUEVO**

En ocasiones será bueno mostrar algunos (pero no todos) de los atributos u operaciones. Para indicar que sólo enseñará algunos de ellos, seguirá la lista de aquellos que mostrará con tres puntos (...), mismos que se conocen como *puntos suspensivos*. A la omisión de ciertos o todos los atributos y operaciones se le conoce como *abreviar* una clase. La figura 3.10 le muestra el uso de los puntos suspensivos.

**FIGURA 3.10**

*Los puntos suspensivos indican atributos u operaciones que no se encuentran en todo el conjunto.*



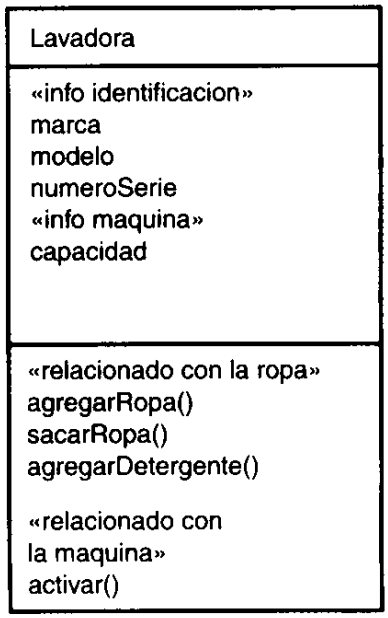

www.elsolucionario.net

www.elsolucionario.net

Si usted tiene una larga lista de atributos u operaciones podrá utilizar un estereotipo para organizarla de forma que sea más comprensible. Un estereotipo es el modo en que el UML le permite extenderlo, es decir, crear nuevos elementos que son específicos de un problema en particular que intente resolver. Como lo mencioné en la hora 1, usted muestra un estereotipo como un nombre bordeado por dos pares de paréntesis angulares. Para una lista de atributos, podrá utilizar un estereotipo como encabezado de un subconjunto de atributos, como en la figura 3.11.

**FIGURA 3.11**

*Podrá usar un estereotipo para organizar una lista de atributos u operaciones.*



El estereotipo es una estructura flexible, la cual podrá utilizar de diversos modos. Por ejemplo, podrá utilizar el estereotipo sobre el nombre de una clase en un símbolo de clase para indicar algo respecto al papel de la clase.

# Responsabilidades y restricciones

TERMINO NUEVO

El símbolo de clase le permite establecer otro tipo de información de sí misma. En un área bajo la lista de operaciones, podrá mostrar la responsabilidad de la clase. La *responsabilidad* es una descripción de lo que hará la clase, es decir, lo que sus atributos y operaciones intentan realizar en conjunto. Una lavadora, por ejemplo, tiene la responsabilidad de recibir ropa sucia y dar por resultado ropa limpia.

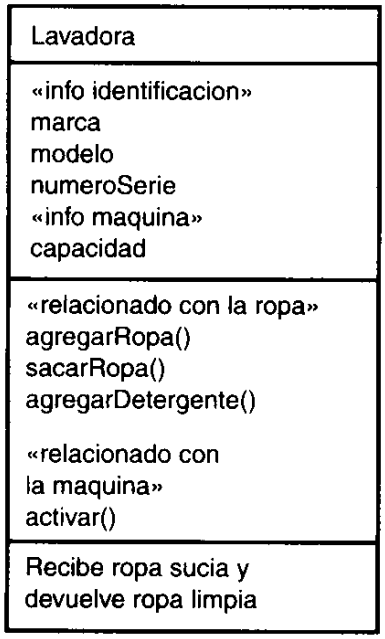
En el símbolo, indicará las responsabilidades en un área inferior a la que contiene las operaciones (vea la figura 3.12).

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 3.12**

*En un símbolo de clase, que irá debajo de la lista de operaciones, escribirá las responsabilidades de la clase.*

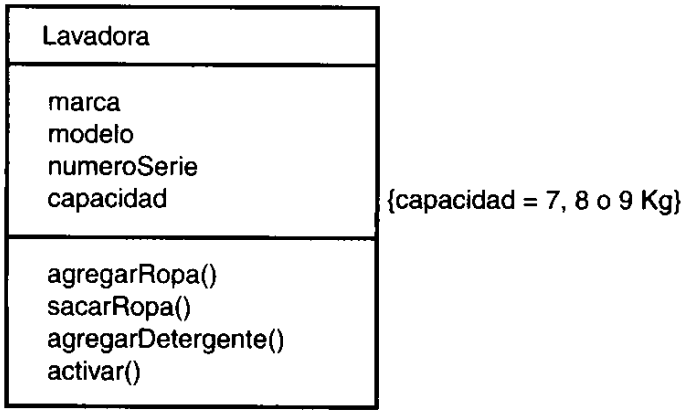



La idea es incluir información suficiente para describir una clase de forma inequívoca.

**TERMINO NUEVO** Una manera más formal es agregar una *restricción*, un texto libre bordeado por llaves; este texto especifica una o varias reglas que sigue la clase. Por ejemplo, suponga que en la clase Lavadora usted desea establecer que la capacidad de una lavadora será de 7, 8 o 9 Kg (y así, "restringir" el atributo capacidad de la clase Lavadora). Usted escribirá {capacidad = 7, 8 o 9 Kg} junto al símbolo de la clase Lavadora. La figura 3.13 le muestra cómo hacerlo.

**FIGURA 3.13**

*La regla entre llaves restringe al atributo capacidad para contener uno de tres posibles valores.*



El UML funciona con otra forma –aún más formal– de agregar restricciones que hacen más explícitas las definiciones. Es todo un lenguaje conocido como *OCL (Lenguaje de restricción de objetos)*. OCL cuenta con su propio conjunto de reglas, términos y operadores, lo que lo convierte en una herramienta avanzada y, en ocasiones, útil.

www.elsolucionario.net

www.elsolucionario.net

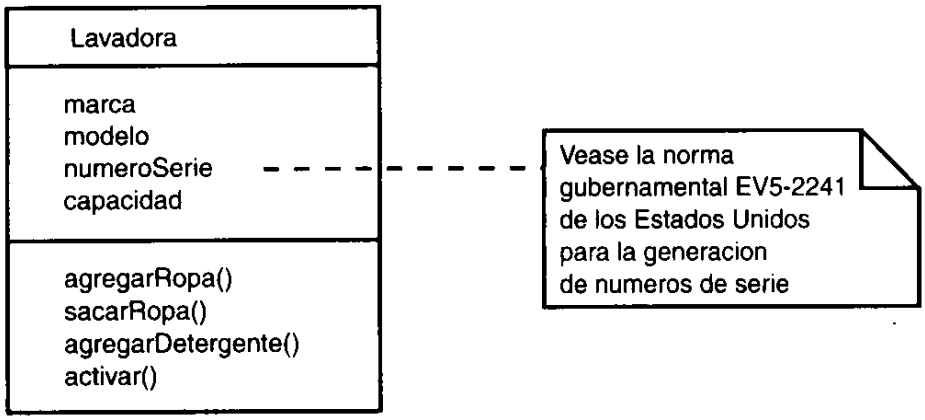
# Notas adjuntas

Por encima y debajo de los atributos, operaciones, responsabilidades y restricciones, puede agregar mayor información a una clase en la figura de notas adjuntas.

Con frecuencia agregará una nota a un atributo u operación. La figura 3.14 le muestra una nota que se refiere a una norma gubernamental que indica dónde encontrar la manera en que se generan los números de serie para los objetos de la clase Lavadora.

**FIGURA 3.14**

*Una nota adjunta proporciona mayor información respecto a la clase.*



Una nota puede contener tanto una imagen como texto.

# Qué es lo que hacen las clases y cómo encontrarlas

Las clases son el vocabulario y terminología de un área del conocimiento. Conforme hable con los clientes, analice su área de conocimiento y diseñe sistemas de computación que resuelvan los problemas de dicha área, comprenderá la terminología y modelará los términos como clases en el UML.

En sus conversaciones con los clientes preste atención a los sustantivos que utilizan para describir las entidades de sus negocios; ya que dichos sustantivos se convertirán en las clases de su modelo. También preste atención a los verbos que escuche, dado que constituirán las operaciones de sus clases. Los atributos surgirán como sustantivos relacionados con los nombres de la clase. Una vez que tenga una lista básica de las clases, pregunte a los clientes qué es lo que cada clase hace dentro del negocio. Sus respuestas le indicarán las responsabilidades de la clase.

Suponga que usted es un analista que generará un modelo del juego de baloncesto, y que entrevista a un entrenador para comprender el juego. La conversación podría surgir como sigue:

www.elsolucionario.net

www.elsolucionario.net

Analista: "Entrenador, ¿de qué se trata el juego?"

Entrenador: "Consiste en arrojar el balón a través de un aro, conocido como cesto, y hacer una mayor puntuación que el oponente. Cada equipo consta de cinco jugadores: dos defensas, dos delanteros y un central. Cada equipo lleva el balón al cesto del equipo oponente con el objetivo de hacer que el balón sea encestado."

Analista: "¿Cómo se hace para llevar el balón al otro cesto?"

Entrenador: "Mediante pases y dribles. Pero el equipo tendrá que encestar antes de que termine el lapso para tirar."

Analista: "¿El lapso para tirar?"

Entrenador: "Así es, son 24 segundos en el baloncesto profesional, 30 en un juego internacional, y 35 en el colegial para tirar el balón luego de que un equipo toma posesión de él."

Analista: "¿Cómo funciona el puntaje?"

Entrenador: "Cada canasta vale dos puntos, a menos que el tiro haya sido hecho detrás de la línea de los tres puntos. En tal caso, serán tres puntos. Un tiro libre contará como un punto. A propósito, un tiro libre es la penalización que paga un equipo por cometer una infracción. Si un jugador infracciona a un oponente, se detiene el juego y el oponente puede realizar diversos tiros al cesto desde la línea de tiro libre."

Analista: "Hábleme más acerca de lo que hace cada jugador."

Entrenador: "Quienes juegan de defensa son, en general, quienes realizan la mayor parte de los dribles y pases. Por lo general tienen menor estatura que los delanteros, y éstos, a su vez, son menos altos que el central (que también se conoce como 'poste'). Se supone que todos los jugadores pueden burlar, pasar, tirar y rebotar. Los delanteros realizan la mayoría de los rebotes y los disparos de mediano alcance, mientras que el central se mantiene cerca del cesto y dispara desde un alcance corto."

Analista: "¿Qué hay de las dimensiones de la cancha? Y ya que estamos en eso ¿cuánto dura el juego?"

Entrenador: "En un juego internacional, la cancha mide 28 metros de longitud y 15 de ancho; el cesto se encuentra a 3.05 m del piso. En un juego profesional, el juego dura 48 minutos, divididos en cuatro cuartos de 12 minutos cada uno. En un juego colegial e internacional, la duración es de 40 minutos, divididos en dos mitades de 20 minutos. Un cronómetro del juego lleva un control del tiempo restante."

La plática podría continuar, pero hagamos una pausa y veamos con qué contamos. Aquí hay varios sustantivos que ha descubierto: balón, cesto, equipo, jugadores, defensas, delanteros, centro (o poste), tiro, lapso para tirar, línea de los tres puntos, tiro libre, infracción, línea de tiro libre, cancha, cronómetro del juego.

Y los verbos: tirar, avanzar, driblar (o burlar), pasar, infraccionar, rebotar. También cuenta con cierta información adicional respecto a algunos de los sustantivos (como las

www.elsolucionario.net

www.elsolucionario.net

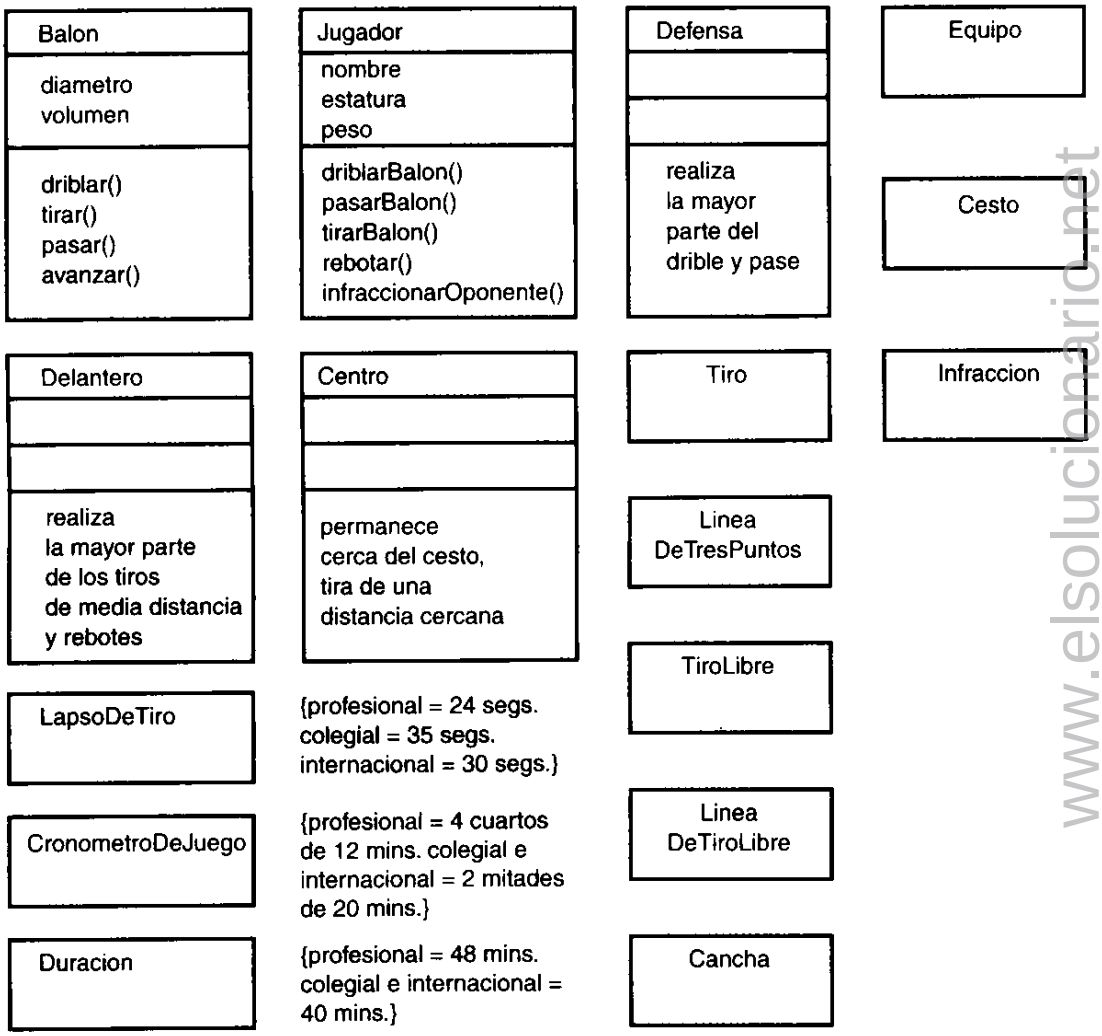
estaturas relativas de los jugadores de cada posición, las dimensiones de la cancha, la cantidad total de tiempo en un lapso de tiro y la duración de un juego).

Finalmente, su propio sentido común podría entrar en acción para generar ciertos atributos por usted mismo. Usted sabe, por ejemplo, que el balón cuenta con ciertos atributos, como volumen y diámetro.

A partir de esta información, podrá crear un diagrama como el de la figura 3.15. En él se muestran las clases y se proporcionan ciertos atributos, operaciones y restricciones. El diagrama también muestra las responsabilidades. Podría usar este diagrama como fundamento para otras conversaciones con el entrenador para obtener mayor información.

**FIGURA 3.15**

*Un diagrama inicial para modelar el juego de baloncesto.*




www.elsolucionario.net

www.elsolucionario.net

# Resumen

Un rectángulo es, en el UML, la representación simbólica de una clase. El nombre, atributos, operaciones y responsabilidades de la clase se colocan en áreas delimitadas dentro del rectángulo. Puede utilizar un estereotipo para organizar las listas de atributos y operaciones y además abreviar una clase al mostrar sólo un subconjunto de sus atributos y operaciones. Esto hace un diagrama de clases menos complejo.

Podrá mostrar el tipo de un atributo, su valor inicial y enseñar los valores con que funciona una operación, así como sus tipos. En una operación, esta información se conoce como firma.

Para reducir la ambigüedad en la descripción de una clase agregue restricciones. El UML también le permite indicar mayor información respecto a una clase mediante notas adjuntas al rectángulo que la representa.

Las clases representan el vocabulario de un área del conocimiento. Las conversaciones con el cliente o un experto en el área dejarán entrever los sustantivos que se convertirán en clases en un modelo, y los verbos se transformarán en operaciones. Podrá utilizar un diagrama de clases como una forma de estimular al cliente a que diga más respecto a su área y que ponga en evidencia cierta información adicional.

# Preguntas y respuestas

**P Usted mencionó el uso del "sentido común" para generar el diagrama de clases del baloncesto. Ello suena bien en tal instancia pero, ¿qué ocurre cuando tengo que analizar un área desconocida para mí (donde el sentido común no será de mucha ayuda)?**

**R** Por lo general, contará con cierto apoyo en un área desconocida para usted. Antes de que se reúna con un cliente o con un experto en el campo, intente convertirse en un "subexperto". Prepárese para la reunión y lea cuanta documentación relacionada tenga a la mano. Pregunte a sus entrevistados respecto a documentos o manuales que hayan escrito. Cuando haya terminado de leer, tendrá cierto conocimiento básico y podrá realizar las preguntas indicadas.

**P ¿En qué momento tendría que mostrar la firma de una operación?**

**R** Tal vez, luego de la fase de análisis de un proceso de desarrollo, conforme se adentre en el diseño. La firma es una sección de información que los desarrolladores podrían encontrar muy útil.

www.elsolucionario.net

www.elsolucionario.net

# Taller

Para repasar lo que ha aprendido respecto a la orientación a objetos, intente responder a las siguientes preguntas. Las respuestas las encontrará en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿Cómo representa una clase en el UML?
2. ¿Qué información puede mostrar en un símbolo de clase?
3. ¿Qué es una restricción?
4. ¿Para qué adjuntaría una nota a un símbolo de clase?

## Ejercicios

1. He aquí una breve (e incompleta) descripción del balompié:

   Un equipo de balompié (o fútbol soccer) consiste en 11 jugadores de campo (1 portero y el resto, jugadores de cancha que, en ocasiones, se organizan en cuatro defensas, tres centrales y tres delanteros). Los jugadores pueden usar cualquier parte de su cuerpo (excepto las manos) para introducir el balón a la portería del equipo contrario. La única excepción a esta regla la tiene el portero, quien puede utilizar también las manos para jugar el balón, pero sólo dentro del área de meta. El campo de juego es un rectángulo de una longitud máxima de 120 m y mínima de 90 m; y con una anchura no mayor de 90 m, ni menor de 45. Para partidos internacionales, la longitud será de 110 m como máximo y 100 como mínimo; y una anchura no superior a 75 m ni inferior a 64. En cualquier caso, deberá ser mayor la longitud que la anchura. El campo de juego se dividirá en dos mitades transversales de igual tamaño. El centro del campo será marcado con un punto visible, alrededor del cual se trazará una circunferencia de 9.15 m de radio. La meta del juego es pasar el balón a los delanteros, quienes están mejor preparados para patear el balón a la portería. El portero (o arquero) es la última línea de defensa que intentará bloquear, con cualquier parte de su cuerpo, los tiros de sus opositores. Cada vez que evita un gol, es decir, que el balón entre a la portería, habrá salvado su meta. Cada gol equivale a un punto. Un juego dura 90 minutos, divididos en dos periodos de 45 minutos cada uno.

   Válgase de la anterior información para crear un diagrama como el de la figura 3.15. Si usted conoce más del balompié de lo que he descrito, agregue tal información a su diagrama.

2. Si usted conoce más del baloncesto de lo que hay en la figura 3.15, agregue la información a tal diagrama.

www.elsolucionario.net

www.elsolucionario.net

# HORA 4

# Uso de relaciones

En la hora anterior creamos un conjunto de clases que representaban el vocabulario del baloncesto. Aunque ello le da las bases para una mayor exploración de lo que es el baloncesto, tal vez haya sentido que algo le falta.

Ese "algo" es un sentido en el que las clases se relacionan entre sí. Si observa el modelo (vea la figura 3.15), verá que no se indica la manera en que un jugador se relaciona con un balón, ni cómo los jugadores conforman un equipo, ni la forma en que procede el juego. Es como si hubiera construido una lista de elementos, en lugar de una representación de un área del conocimiento. Es importante saber cómo se conectan las clases entre sí.

Ahora trazaremos las conexiones entre las clases y completaremos la representación.

En esta hora se tratarán los siguientes temas:

- Asociaciones
- Multiplicidad
- Asociaciones calificadas

www.elsolucionario.net

www.elsolucionario.net

- Asociaciones reflexivas
- Herencia y generalización
- Dependencias

# Asociaciones

**TERMINO NUEVO** Cuando las clases se conectan entre sí de forma conceptual, esta conexión se conoce como *asociación*. El modelo inicial de baloncesto le dará algunos ejemplos. Examinemos uno de ellos: la asociación entre un jugador y un equipo. Podrá caracterizar tal asociación con la frase: "un jugador participa en un equipo". Visualizará la asociación como una línea que conectará a ambas clases, con el nombre de la asociación ("participa en") justo sobre la línea. Es útil indicar la dirección de la relación, y lo hará con un triángulo relleno que apunte en la dirección apropiada. La figura 4.1 le muestra cómo visualizar la asociación "Participa en" entre el jugador y el equipo.

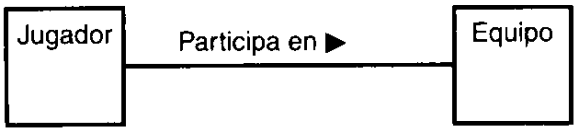



**FIGURA 4.1**
*Una asociación entre un jugador y un equipo.*

Cuando una clase se asocia con otra, cada una de ellas juega un papel dentro de tal asociación. Puede representar estos papeles en el diagrama escribiéndolos cerca de la línea que se encuentra junto a la clase que juega el papel correspondiente. En la asociación entre un jugador y un equipo, si el equipo es profesional, éste es un empleador y el jugador es un empleado. La figura 4.2 le muestra cómo representar dichos papeles.

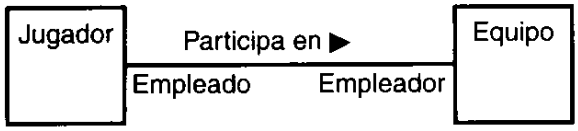



**FIGURA 4.2**
*Por lo general, en una asociación cada clase juega un papel. Puede representar tales papeles en el diagrama.*

La asociación puede funcionar en dirección inversa: un equipo emplea a jugadores. Podrá mostrar ambas asociaciones en el mismo diagrama con un triángulo relleno que indique la dirección de cada asociación, como en la figura 4.3.

www.elsolucionario.net

www.elsolucionario.net

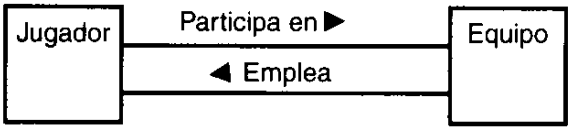



**FIGURA 4.3**

*Pueden aparecer dos asociaciones entre clases en el mismo diagrama.*

Las asociaciones podrían ser más complejas que tan sólo una clase conectada a otra. Varias clases se pueden conectar a una. Si toma en cuenta los defensas, delanteros y central, así como sus asociaciones con la clase Equipo, tendrá el diagrama de la figura 4.4.

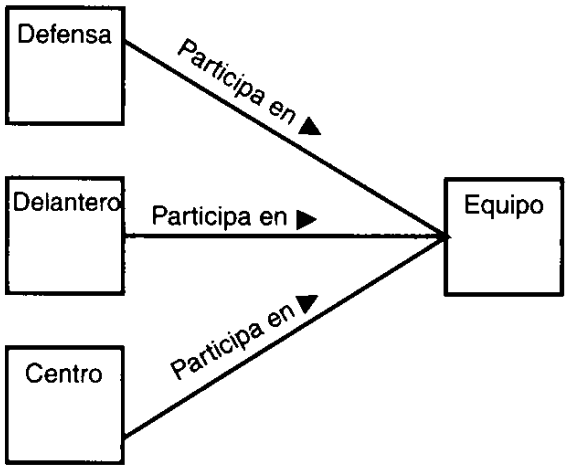



**FIGURA 4.4**

*Pueden asociarse diversas clases con una en particular.*

## Restricciones en las asociaciones

En ocasiones una asociación entre dos clases debe seguir cierta regla. Ésta se indica al establecer una restricción junto a la línea de asociación. Por ejemplo: un Cajero atiende a un Cliente, pero cada Cliente es atendido en el orden en que se encuentre en la formación. Puede capturar este modelo colocando la palabra *ordenado* entre llaves (para indicar la restricción) junto a la clase Cliente, como se ve en la figura 4.5.

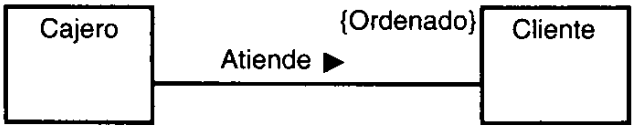



**FIGURA 4.5**

*Puede establecer una restricción en una asociación. En este caso, la asociación Atiende está restringida para que el Cajero atienda al Cliente en turno.*

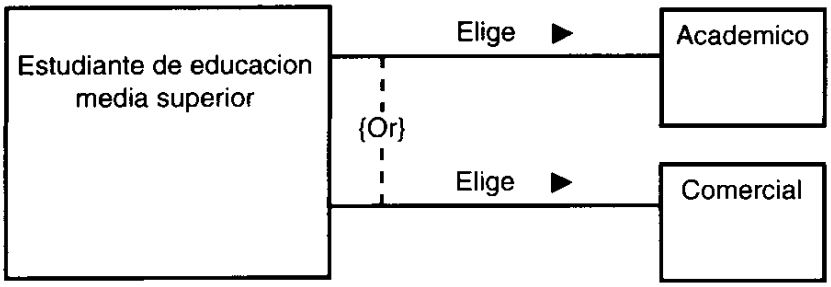
Otro tipo de restricción es la relación O (distinguida como {Or}) en una línea discontinua que conecte a dos líneas de asociación. La figura 4.6 modela a un estudiante de educación media superior que elegirá entre un curso académico o uno comercial.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 4.6**

*La relación O entre dos asociaciones en una restricción.*



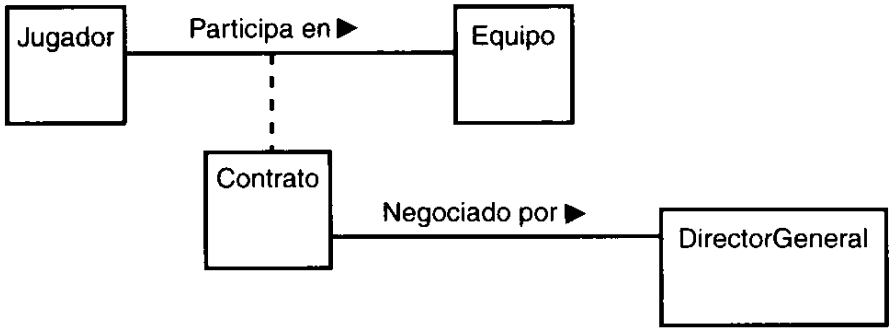

## Clases de asociación

TÉRMINO NUEVO

Una asociación, al igual que una clase, puede contener atributos y operaciones. De hecho, cuando éste sea el caso, usted tendrá una *clase de asociación*. Puede concebir a una clase de asociación de la misma forma en que lo haría con una clase estándar, y utilizará una línea discontinua para conectarla a la línea de asociación. Una clase de asociación puede tener asociaciones con otras clases. La figura 4.7 le muestra una clase de asociación para la asociación "Participa en" entre un jugador y un equipo. La clase de asociación, Contrato, se asocia con la clase DirectorGeneral.

**FIGURA 4.7**

*Una clase de asociación modela los atributos y operaciones de una asociación. Se conecta a una asociación mediante una línea discontinua, y puede asociarse a otra clase.*

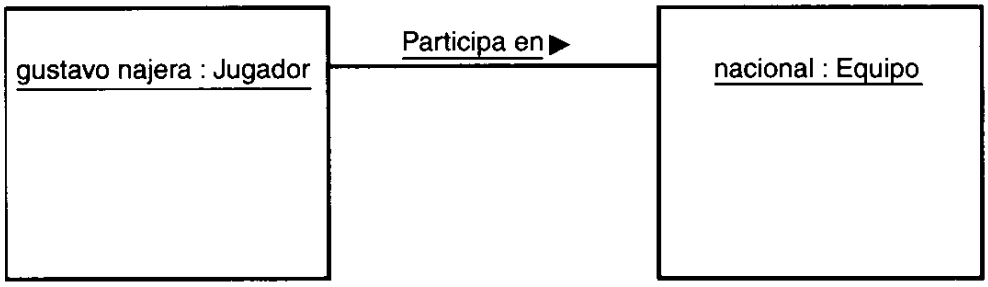



## Vínculos

Así como un objeto es una instancia de una clase, una asociación también cuenta con instancias. Si podemos imaginar a un jugador específico que juega para un equipo específico, la relación "Participa en" se conocerá como *vínculo*, y usted lo representará como una línea que conecta a dos objetos. Tal como tuvo que subrayar el nombre de un objeto, deberá subrayar el nombre de un vínculo, como en la figura 4.8.

**FIGURA 4.8**

*Un vínculo es la instancia de una asociación. Conecta a los objetos en lugar de las clases. Deberá subrayar el nombre del vínculo, como se hace en el nombre de un objeto.*




www.elsolucionario.net

# Multiplicidad

La asociación trazada entre Jugador y Equipo sugiere que las dos clases tienen una relación de uno a uno. No obstante, el sentido común nos indica que éste no es el caso. Un equipo de baloncesto cuenta con cinco jugadores (sin contar a los sustitutos). La asociación Tiene (Has) debe participar en este recuento. En la otra dirección, un jugador puede participar sólo en un equipo, y la asociación "Participa en" debe responder de esto.

**TÉRMINO NUEVO**

Tales especificaciones son ejemplos de la *multiplicidad*: la cantidad de objetos de una clase que se relacionan con un objeto de la clase asociada. Para representar los números en el diagrama, los colocará sobre la línea de asociación junto a la clase correspondiente, como se denota en la figura 4.9.

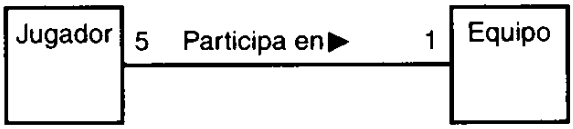



**FIGURA 4.9**

*La multiplicidad señala la cantidad de objetos de una clase que pueden relacionarse con un objeto de una clase asociada.*

La multiplicidad de este ejemplo no es la única que existe. Hay varios tipos de multiplicidades (una multiplicidad de multiplicidades, por decirlo así). Una clase puede relacionarse con otra en un esquema de uno a uno, uno a muchos, uno a uno o más, uno a ninguno o uno, uno a un intervalo definido (por ejemplo: uno a cinco hasta diez), uno a exactamente *n* (como en este ejemplo), o uno a un conjunto de opciones (por ejemplo, uno a nueve o diez). El UML utiliza un asterisco (*) para representar *más* y para representar *muchos*. En un contexto O se representa por dos puntos, como en "1..*" ("uno o más"). En otro contexto, O se representa por una coma, como en "5, 10" ("5 o 10"). La figura 4.10 le muestra cómo concebir las posibles multiplicidades.

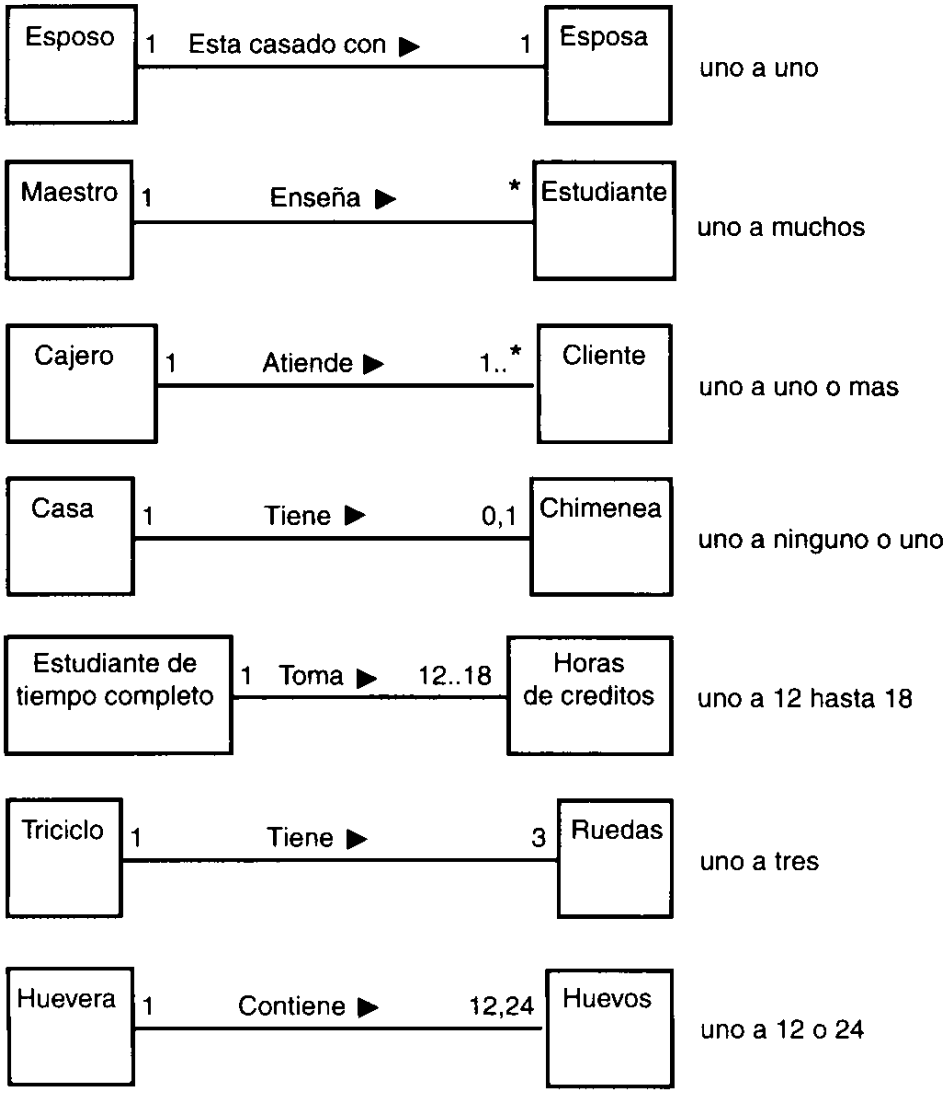
Cuando la clase A tiene una multiplicidad de uno a ninguno o uno con la clase B, la clase B se dice que es opcional para la clase A.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 4.10**

*Posibles multiplicidades y cómo representarlas en el UML.*




# Asociaciones calificadas

Cuando la multiplicidad de una asociación es de uno a muchos, con frecuencia se presenta un reto muy particular: la búsqueda. Cuando un objeto de una clase tiene que seleccionar un objeto particular de otro tipo para cumplir con un papel en la asociación, la primera clase deberá atenerse a un atributo en particular para localizar al objeto adecuado. Normalmente, dicho atributo es un identificador que puede ser un número de identidad. Por ejemplo, cuando usted realiza una reservación en un hotel, el hotel le asigna un número de confirmación. Si usted quiere hacer preguntas respecto a la reservación, deberá proporcionar el número de confirmación.

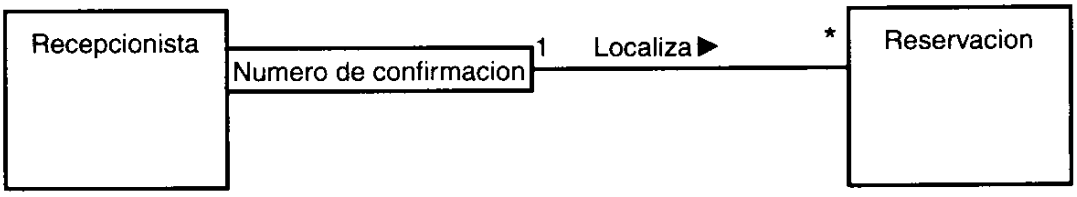
**TÉRMINO NUEVO** En el UML la información de identidad se conoce como *calificador*. Su símbolo es un pequeño rectángulo adjunto a la clase que hará la búsqueda. La figura 4.11 muestra la representación. La idea es reducir, con eficiencia, la multiplicidad de uno a muchos a una multiplicidad de uno a uno.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 4.11**

*Un calificador en una asociación resuelve el problema de la búsqueda.*

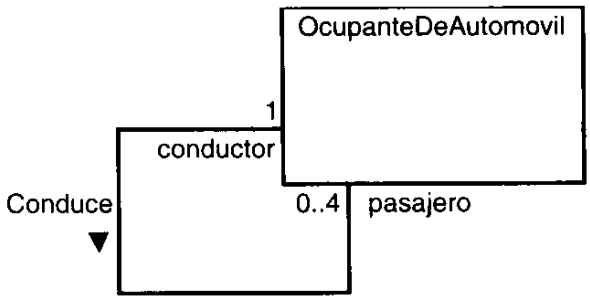



# Asociaciones reflexivas

En ocasiones, una clase es una asociación consigo misma. Esto puede ocurrir cuando una clase tiene objetos que pueden jugar diversos papeles. Un OcupanteDeAutomovil puede ser un Conductor o un Pasajero. En el papel del conductor, el OcupanteDeAutomovil puede llevar ninguno o más OcupanteDeAutomovil, quienes jugarán el papel de pasajeros. Esto lo representará mediante el trazado de una línea de asociación a partir del rectángulo de la clase hacia el mismo rectángulo de la clase, y en la línea de asociación indicará los papeles, nombre de la asociación, dirección de la asociación y multiplicidad como ya lo hizo antes. La figura 4.12 le presenta este ejemplo.

**FIGURA 4.12**

*En una asociación reflexiva, trazará la línea de la clase hacia sí misma y podrá incluir los papeles, nombre de la asociación y su dirección, así como su multiplicidad.*




# Herencia y generalización

Uno de los sellos distintivos de la orientación a objetos es que captura uno de los mayores aspectos del sentido común en cuanto a la vida diaria: si usted conoce algo de una categoría de cosas, automáticamente sabrá algunas cosas que podrá transferir a otras categorías. Si usted sabe que algo es un electrodoméstico, ya sabrá que contará con un interruptor, una marca y un número de serie. Si sabe que algo es un animal dará por hecho que come, duerme, tiene una forma de nacer, de trasladarse de un lugar a otro y algunos otros atributos (y operaciones) que podría listar si pensara en ello por algunos instantes.

**TÉRMINO NUEVO**

La orientación a objetos se refiere a esto como *herencia*. El UML también lo denomina *generalización*. Una clase (la clase secundaria o subclase) puede heredar los atributos y operaciones de otra (la clase principal o superclase). La clase principal (o madre) es más genérica que la secundaria (o hija).

www.elsolucionario.net

www.elsolucionario.net

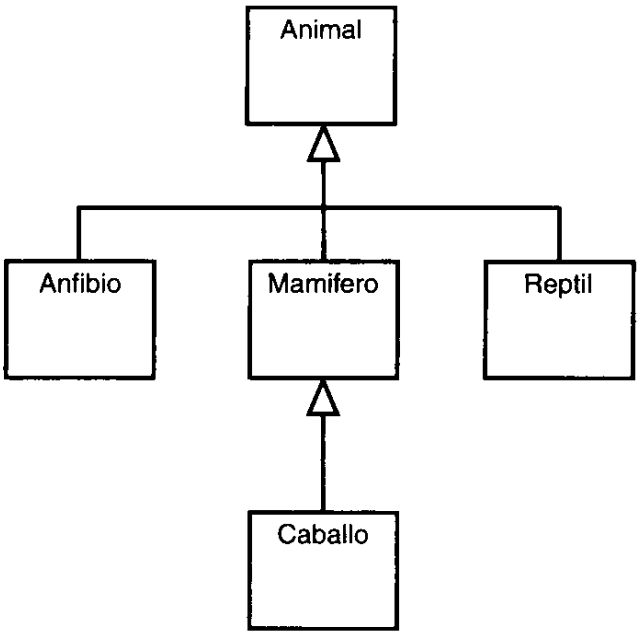
En la generalización, una clase secundaria (hija) es sustituible por una clase principal (madre). Es decir, donde quiera que se haga referencia a la clase madre, también se hace referencia a la clase hija. Sin embargo, en el caso contrario no es aplicable.

La jerarquía de la herencia no tiene que finalizar en dos niveles: una clase secundaria puede ser principal para otra clase secundaria. Un Mamífero es una clase secundaria de Animal, y Caballo es una clase secundaria de Mamífero.

En el UML representará la herencia con una línea que conecte a la clase principal con la secundaria. En la parte de la línea que se conecta con la clase principal, colocará un triángulo sin rellenar que apunte a la clase principal. Este tipo de conexión se interpreta con la frase *es un tipo de*. Un Mamífero *es un tipo de* Animal, y un Caballo *es un tipo de* Mamífero. La figura 4.13 le muestra esta particular jerarquía de la herencia, junto con otras clases. Observe la apariencia del triángulo y las líneas cuando varias clases secundarias son herencia de una clase principal. Al disponer el diagrama de este modo, trae por resultado un diagrama más ordenado en lugar de mostrar todas las líneas y triángulos, aunque el UML no le prohíbe colocarlos todos en la imagen. También vea que no colocó los atributos y operaciones heredadas en los rectángulos de las subclases, dado que ya los había representado en la superclase.

**FIGURA 4.13**

*Una jerarquía de herencia en el reino animal.*



Cuando modele la herencia, tenga la seguridad de que la clase secundaria satisfaga la relación *es un tipo de* con la clase principal. Si no se cumple tal relación, tal vez una asociación de otro tipo podría ser más adecuada.

www.elsolucionario.net

www.elsolucionario.net

Con frecuencia las clases secundarias agregan otras operaciones y atributos a los que han heredado. Por ejemplo: un Mamífero tiene pelo y da leche, dos atributos que no se encuentran en la clase Animal.

TERMINO NUEVO

Una clase puede no provenir de una clase principal, en cuyo caso será una *clase base* o *clase raíz*. Una clase podría no tener clases secundarias, en cuyo caso será una clase final o clase hoja. Si una clase tiene exactamente una clase principal, tendrá una *herencia simple*. Si proviene de varias clases principales, tendrá una *herencia múltiple*.

## Descubrimiento de la herencia

En el proceso de plática con un cliente, un analista descubrirá la herencia de varias formas. Es posible que las clases candidatas que aparezcan incluyan tanto clases principales como clases secundarias. El analista deberá darse cuenta que los atributos y operaciones de una clase son generales y que se aplicarán a, quizá, varias clases (mismas que agregarán sus propios atributos y operaciones).

El ejemplo del baloncesto de la hora 3, "Uso de la orientación a objetos", tiene las clases Jugador, Defensa, Delantero y Central. El Jugador tiene atributos como nombre, estatura, peso, velocidadAlCorrer y saltoVertical. Tiene operaciones como driblar(), pasar(), rebotar() y tirar(). Las clases Defensa, Delantero y Centro heredarán tales atributos y operaciones, y agregarán los suyos. La clase Defensa podría tener las operaciones correrAlFrente() y quitarBalon(). El Central podría tener retacarBalon(). De acuerdo con los comentarios del entrenador respecto a las estaturas de los jugadores, el analista tal vez quisiera colocar restricciones en las estaturas para cada posición.

Otra posibilidad es que el analista note que dos o más clases tienen ciertos atributos y operaciones en común. El modelo del baloncesto tiene un CronometroDeJuego (que controla el tiempo que resta en un periodo de juego) y un LapsoDeTiro (que controla el tiempo restante desde el instante que un equipo tomó posesión del balón, hasta que intente encestar). Si nos damos cuenta de que ambos controlan el tiempo, el analista podría formular una clase Reloj con una operación controlarTiempo() que podrían heredar tanto CronometroDeJuego como LapsoDeTiro.

Dado que LapsoDeTiro controla 24 segundos (profesional) o 35 segundos (colegial) y el CronometroDeJuego controla 12 minutos (profesional) o 20 minutos (colegial), controlarTiempo() será polimórfico.

## Clases abstractas

En el modelo del baloncesto, el par de clases que mencioné —Jugador y Reloj— son útiles puesto que funcionan como clases principales para clases secundarias importantes. Las clases secundarias son importantes en el modelo dado que finalmente usted querrá

www.elsolucionario.net

www.elsolucionario.net

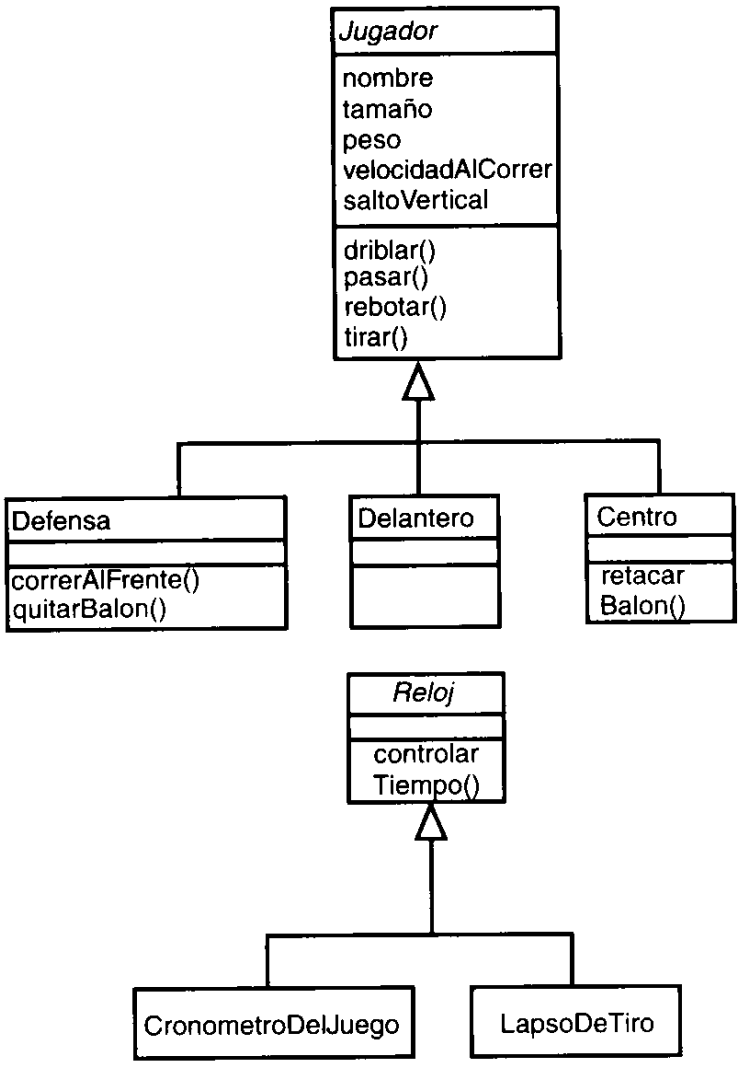
tener instancias de tales clases. Para desarrollar el modelo, necesitará instancias de Defensa, Delantero, Centro, CronometroDeJuego y LapsoDeTiro.

No obstante, Jugador y Reloj no proporcionan ninguna instancia al modelo. Un objeto de la clase Jugador no serviría a ningún propósito, así como tampoco uno de la clase Reloj.

**TÉRMINO NUEVO** Las clases como Jugador y Reloj —que no proveen objetos— se dice que son *abstractas*. Una clase abstracta se distingue por tener su nombre en cursivas. La figura 4.14 muestra las dos clases abstractas y sus clases secundarias.

**FIGURA 4.14**

*Dos jerarquías de herencia con clases abstractas en el modelo de baloncesto.*




# Dependencias

**TÉRMINO NUEVO** En otro tipo de relación, una clase utiliza a otra. A esto se le llama dependencia. El uso más común de una dependencia es mostrar que la firma de la operación de una clase utiliza a otra clase.

Suponga que diseñará un sistema que muestra formularios corporativos en pantalla para que los empleados los llenen. El empleado utiliza un menú para seleccionar el formulario por llenar. En su diseño, tiene una clase Sistema y una clase Formulario. Entre sus

www.elsolucionario.net

www.elsolucionario.net

muchas operaciones, la clase Sistema tiene mostrarFormulario(f:Form). El formulario que el sistema desplegará, dependerá, obviamente, del que elija el usuario. La notación del UML para ello es una línea discontinua con una punta de flecha en forma de triángulo sin relleno que apunta a la clase de la que depende, como muestra la figura 4.15.

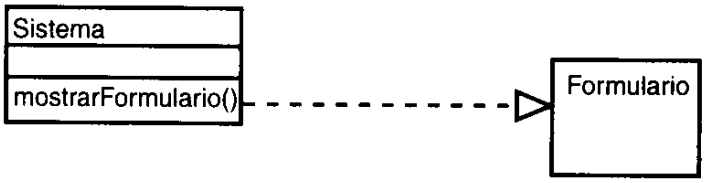



**FIGURA 4.15**

*Una flecha representada por una línea discontinua con una punta de flecha en forma de triángulo sin relleno simboliza una dependencia.*

# Resumen

Sin las relaciones, un modelo de clases sería poco menos que una lista de cosas que representarían un vocabulario. Las relaciones le muestran cómo se conectan los términos del vocabulario entre sí para dar una idea de la sección del mundo que se modela. La asociación es la conexión conceptual fundamental entre clases. Cada clase en una asociación juega un papel, y la multiplicidad especifica cuántos objetos de una clase se relacionan con un objeto de la clase asociada. Hay muchos tipos de multiplicidad. Una asociación se representa como una línea entre los rectángulos de clases con los papeles y multiplicidades en cada extremo. Al igual que una clase, una asociación puede contener atributos y operaciones.

Una clase puede heredar atributos y operaciones de otra clase. La clase heredada es secundaria de la clase principal que es de la que se hereda. Descubrirá la herencia cuando encuentre clases en su modelo inicial que tengan atributos y operaciones en común. Las clases abstractas sólo se proyectan como bases de herencia y no proporcionan objetos por sí mismas. La herencia se representa como una línea entre la clase principal y la secundaria, con un triángulo sin rellenar que se adjunta (y apunta a) la clase principal.

En una dependencia, una clase utiliza a otra. El uso más común de una dependencia es mostrar que una firma en la operación de una clase utiliza a otra clase. Una dependencia se proyecta como una línea discontinua que reúne a las dos clases en la dependencia, con una punta de flecha en forma de triángulo sin relleno que adjunta (y apunta a) la clase de la que se depende.

# Preguntas y respuestas

**P ¿En alguna ocasión se le puede poner nombre a una relación de herencia, como se hace en una asociación?**

**R** El UML no le impide que adjudique un nombre a una relación de herencia, pero por lo general esto no es necesario.

www.elsolucionario.net

www.elsolucionario.net

# Taller

El cuestionario y los ejercicios se han diseñado para reafirmar su conocimiento del UML en el área de las relaciones. Cada pregunta y ejercicio requiere que usted piense en la simbología del modelado que ha aprendido y la aplique a una situación. Las respuestas se encuentran en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionarios

1. ¿Cómo representaría la multiplicidad?
2. ¿Cómo descubrirá la herencia?
3. ¿Qué es una clase abstracta?
4. ¿Cuál es el efecto de un calificador?

## Ejercicios

1. Tome como base el modelo del baloncesto de la Hora 3, y agregue vínculos que expresen las relaciones que ha visto en esta hora. Si conoce el juego del baloncesto, siéntase con libertad de agregar los vínculos que representen su conocimiento.
2. De acuerdo con un viejo adagio: "Un abogado que se defiende a sí mismo, tiene por cliente a un tonto." Cree un modelo que refleje esta pieza de sabiduría.

www.elsolucionario.net

www.elsolucionario.net

# HORA 5

# Agregación, composición, interfaces y realización

Continuaremos con las relaciones entre clases y comprenderá nuevos conceptos respecto a las clases y sus diagramas.

En esta hora se tratarán los siguientes temas:

- Agregaciones
- Composiciones
- Contextos
- Interfaces y realizaciones
- Visibilidad

www.elsolucionario.net

www.elsolucionario.net

Ya ha visto lo concerniente a asociación, multiplicidad y herencia y está casi listo para crear diagramas de clases significativos. Conforme explore otros tipos de relaciones y detalles relacionados con las clases comprenderá las piezas finales del rompecabezas. La meta final es crear una idea estática de un sistema, con todas las conexiones entre las clases que lo conforman.

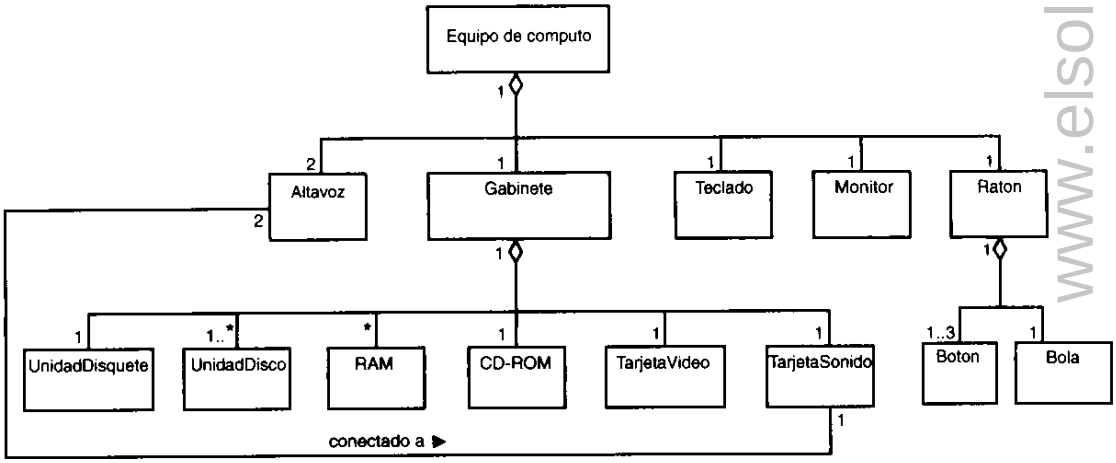
# Agregaciones

**TÉRMINO NUEVO** En ocasiones una clase consta de otras clases. Éste es un tipo especial de relación conocida como *agregación* o *acumulación*. Los componentes y la clase que constituyen son una asociación que conforma un todo. En la hora 2, "Orientación a objetos", mencioné que su computadora es un conjunto de elementos que consta de gabinete, teclado, ratón, monitor, unidad de CD-ROM, una o varias unidades de disco duro, módem, unidad de disquete, impresora y, posiblemente, altavoces. Además de las unidades de disco, el gabinete contiene la memoria RAM, una tarjeta de vídeo y una tarjeta de sonido (tal vez algunos otros elementos).

Puede representar una agregación como una jerarquía dentro de la clase completa (por ejemplo el sistema computacional) en la parte superior, y los componentes por debajo de ella. Una línea conectará el todo con un componente mediante un rombo sin relleno que se colocará en la línea más cercana al todo. La figura 5.1 le muestra el sistema de cómputo como una agregación.

**FIGURA 5.1**

*Una asociación por agregación se representa por una línea entre el componente y el todo con un rombo sin relleno que conforma al todo.*




Aunque este ejemplo le muestra cada componente correspondiente a un todo, en una agregación éste no será necesariamente el caso. Por ejemplo: en un sistema casero de entretenimiento, un control remoto podría ser un componente de una televisión, aunque también podría ser un componente de una reproductora de casetes de vídeo.

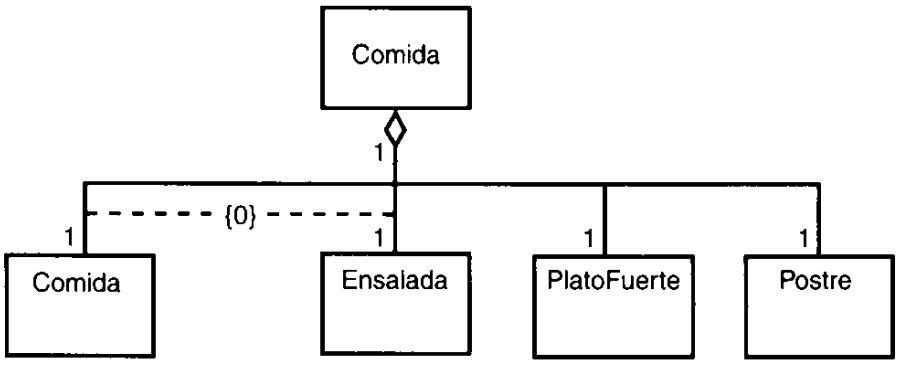
www.elsolucionario.net

## Restricciones en las agregaciones

En ocasiones el conjunto de componentes posibles en una agregación se establece dentro de una relación O. En ciertos restaurantes, una comida consta de sopa o ensalada, el plato fuerte y el postre. Para modelar esto, utilizaría una restricción: la palabra O dentro de llaves con una línea discontinua que conecte las dos líneas que conforman al todo, como lo muestra la figura 5.2.

**FIGURA 5.2**

*Puede establecer una restricción a una agregación para mostrar que un componente u otro es parte del todo.*

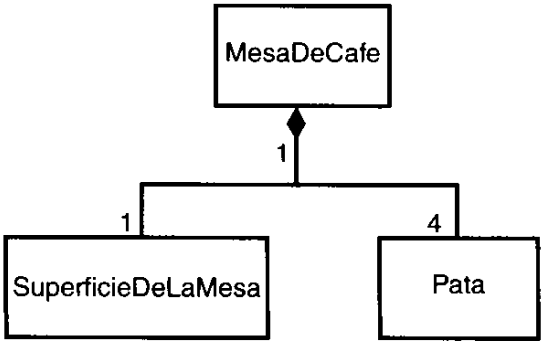



# Composiciones

Una composición es un tipo muy representativo de una agregación. Cada componente dentro de una composición puede pertenecer tan sólo a un todo. Los componentes de una mesa de café (la superficie de la mesa y las patas) establecen una composición. El símbolo de una composición es el mismo que el de una agregación, excepto que el rombo está relleno (vea la figura 5.3).

**FIGURA 5.3**

*En una composición, cada componente pertenece solamente a un todo. Un rombo relleno representa esta relación.*




# Contextos

Cuando modele un sistema podrían producirse, con frecuencia, agrupamientos de clases, como agregaciones o composiciones. En tal caso, deberá enfocar su atención en un agrupamiento o en otro, y el diagrama de contexto le proporciona la característica de modelaje que requiere para tal fin. Las composiciones figuran en gran medida dentro de los diagramas de contexto. Un diagrama de contexto es como un mapa detallado de alguna sección de un mapa de mayores dimensiones. Pueden ser necesarias varias secciones para capturar toda la información detallada.

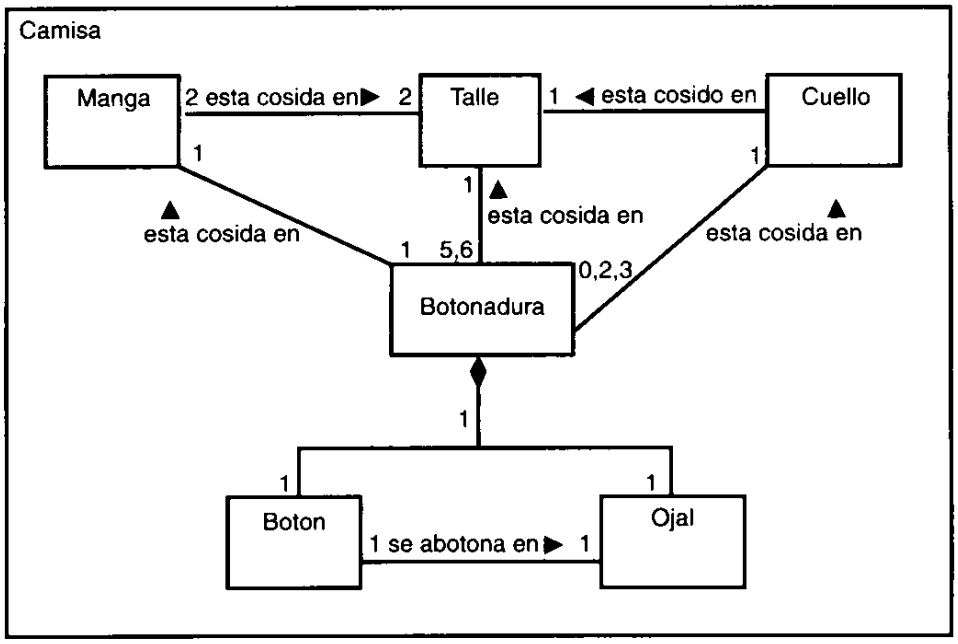
www.elsolucionario.net

www.elsolucionario.net

He aquí un ejemplo: suponga que está creando un modelo de una camisa y la forma en que se podría combinar con algún atuendo y un guardarropa. Un tipo de diagrama de contexto (vea la figura 5.4) le mostrará la camisa como un gran rectángulo de clase, con un diagrama anidado en el interior, el cual le muestra cómo los componentes de la camisa están relacionados entre sí. Éste es un diagrama de contexto de composición (dado que la sola camisa reúne a cada componente se le denomina *de composición*).

**FIGURA 5.4**

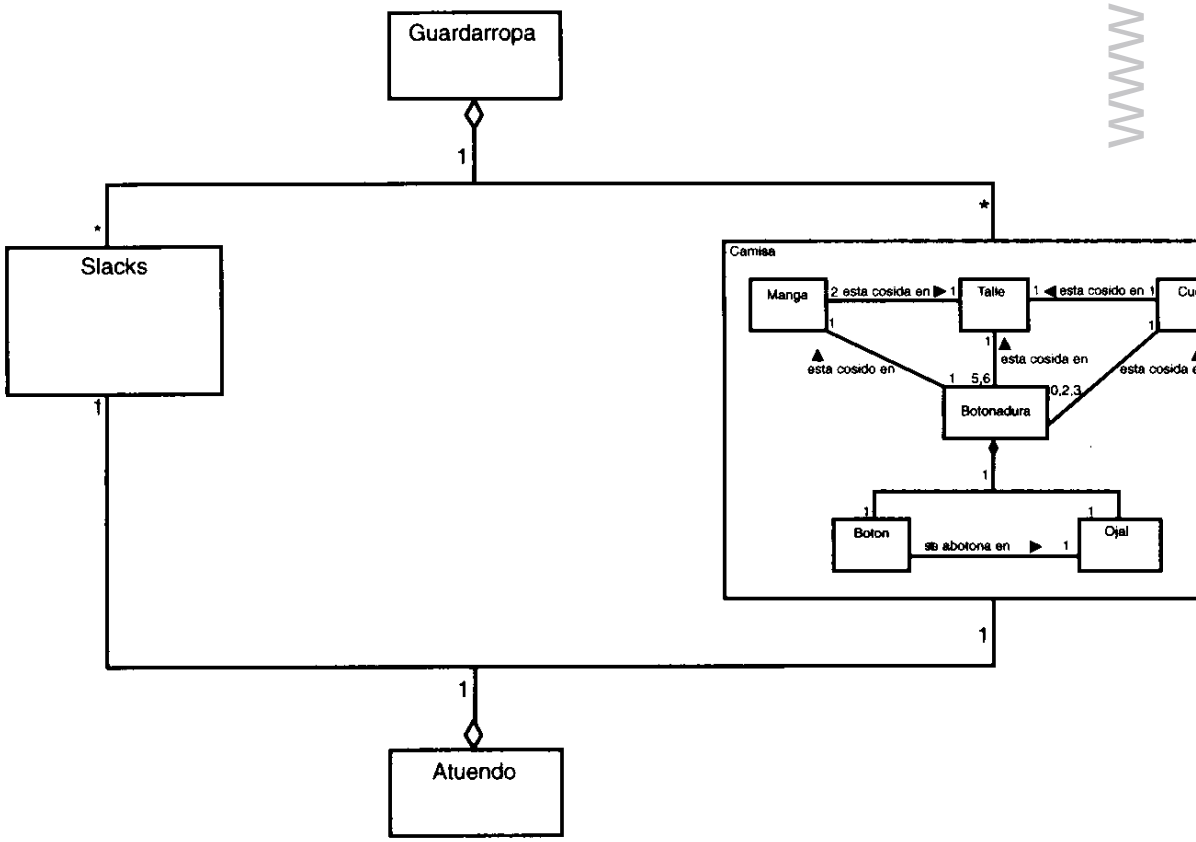
*Un diagrama de contexto de composición le muestra los componentes de una clase como un diagrama anidado dentro de un enorme rectángulo de clase.*




El diagrama de contexto de composición enfoca la atención en la camisa y sus componentes. Para mostrar la camisa en el contexto del guardarropa y de algún atuendo, tendrá que ampliar su ámbito. Un diagrama de contexto del sistema lo hará por usted. Podrá mostrar la forma en que la clase Camisa se conecta con las clases Guardarropa y Atuendo, como se ve en la figura 5.5.

**FIGURA 5.5**

*Un diagrama de contexto del sistema le muestra los componentes de una clase y la forma en que la clase se relaciona con las otras que hay en el sistema.*




www.elsolucionario.net

www.elsolucionario.net

Podrá ver de cerca alguna otra clase y presentar sus detalles en algún otro diagrama de contexto.

# Interfaces y realizaciones

Una vez que haya creado varias clases, tal vez se dé cuenta que no pertenecen a una clase principal, pero en su comportamiento debe incluir algunas de las mismas operaciones con las mismas firmas de la primera clase. Podría codificar las operaciones en una de las clases y reutilizarlas en otras. Una segunda posibilidad es que desarrolle una serie de operaciones para las clases en un sistema, y reutilizarlas para las clases de otro sistema.

TERMINO NUEVO

De cualquier manera, deseará contar con algún medio para capturar el conjunto reutilizable de operaciones. La interfaz es la estructura del UML que le permite hacerlo. Una *interfaz* es un conjunto de operaciones que especifica cierto aspecto de la funcionalidad de una clase, y es un conjunto de operaciones que una clase presenta a otras.

Con un ejemplo podríamos aclarar lo anterior. El teclado que usted utiliza para comunicarse con su equipo es una interfaz reutilizable. Su operación basada en la opresión de teclas ha provenido de la máquina de escribir. La disposición de las teclas es casi la misma que en una máquina de escribir, pero el punto principal es que la operación por opresión de teclas ha sido cedida de un sistema a otro. Otras operaciones (Mayús, Bloq Mayús y Tab) también se integraron a partir de la máquina de escribir.

Por supuesto, el teclado de una computadora incluye diversas operaciones que no encontrará en una máquina de escribir: Control, Alt, RePág, AvPág y otras. Así pues, la interfaz puede establecer un subconjunto de las operaciones de una clase y no necesariamente todas ellas.

Puede modelar una interfaz del mismo modo en que modelaría una clase, con un símbolo rectangular. La diferencia será que, como un conjunto de operaciones, una interfaz no tiene atributos. Recordará que puede omitir los atributos de la representación de una clase. ¿Entonces, cómo distinguiría entre una interfaz y una clase que no muestra sus atributos? Una forma es utilizar la estructura "estereotipo" y especificar la palabra «interfaz» sobre el nombre de la interfaz en el rectángulo. Otra es colocar la letra "I" al principio del nombre de una interfaz.

TERMINO NUEVO

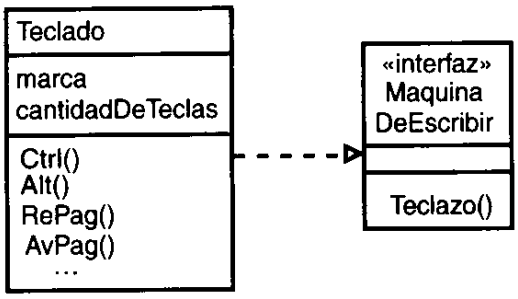
En cierto sentido, es como si el teclado de la computadora garantizara que esta parte de su funcionalidad "haría las veces" del teclado de una máquina de escribir. Bajo este esquema, la relación entre una clase y una interfaz se conoce como *realización*. Esta relación está modelada como una línea discontinua con una punta de flecha en forma de triángulo sin rellenar que adjunte y apunte a la interfaz. La figura 5.6 le muestra cómo se lleva a cabo esto.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 5.6**

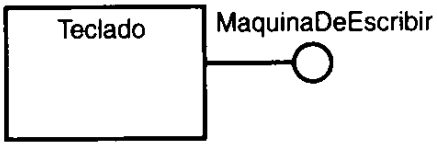
*Una interfaz es un conjunto de operaciones que realiza una clase. Esta última se relaciona con una interfaz mediante la realización, misma que se indica por una línea discontinua con una punta de flecha en forma de triángulo sin rellenar que apunte a la interfaz.*




Otra forma (omitida) de representar una clase y su interfaz es con un pequeño círculo que se conecte mediante una línea a la clase, como se ve en la figura 5.7.

**FIGURA 5.7**

*La forma omitida de representar una clase que realice una interfaz.*




Una clase puede realizar más de una interfaz, y una interfaz puede ser realizada por más de una clase.

## Visibilidad

**TERMINO NUEVO**

El concepto de visibilidad está muy relacionado con las interfaces y la realización. La *visibilidad* se aplica a atributos u operaciones, y establece la proporción en que otras clases podrán utilizar los atributos y operaciones de una clase dada (o en operaciones de una interfaz). Existen tres niveles de visibilidad: Nivel *público*, en el cual la funcionalidad se extiende a otras clases. En el nivel *protegido* la funcionalidad se otorga sólo a las clases que se heredan de la clase original. En el nivel *privado* sólo la clase original puede utilizar el atributo u operación. En una televisión, modificarVolumen() y cambiarCanal() son operaciones públicas, en tanto que dibujarImagenEnPantalla() es privada. En un automóvil, acelerar() y frenar() son operaciones públicas, pero actualizarKilometraje() o actualizarMillaje() es protegida.

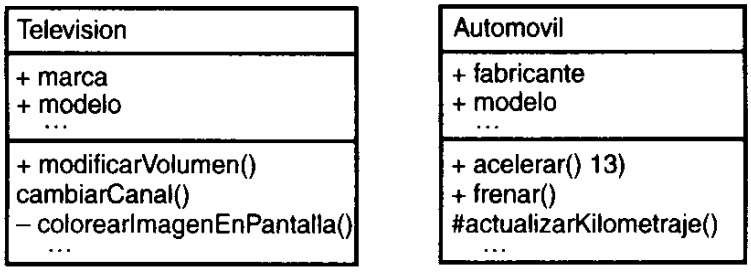
La realización, como podría imaginar, implica que el nivel público se aplique a cualquier operación en una interfaz. La protección de operaciones mediante cualquiera de los otros niveles tal vez no tendría sentido, dado que una interfaz se orienta a ser realizada por diversas clases.

Para indicar el nivel público, anteceda el atributo u operación con un signo de suma (+), para revelar un nivel protegido, antecédalo con un símbolo de número (#), y para indicar el nivel privado, antecédalo con un guión (-). La figura 5.8 muestra los atributos y operaciones públicos, protegidos y privados tanto en una televisión como en un automóvil.

www.elsolucionario.net

**FIGURA 5.8**

*Los atributos y operaciones públicos y privados, tanto de una televisión como de un automóvil.*




## Ámbito

TERMINO NUEVO

El ámbito es otro concepto referente a los atributos y operaciones, y la forma en que se relacionan dentro de un sistema. Hay dos tipos de ámbitos, el de instancia y el de archivador. En el primero cada instancia cuenta con su propio valor en un atributo u operación. En un ámbito de *archivado*, sólo habrá un valor del atributo u operación en todas las instancias de la clase. Un atributo u operación con el ámbito de archivador, aparece con su nombre subrayado. Este tipo de ámbito se utiliza con frecuencia cuando un grupo específico de instancias (ningunas otras) tienen que compartir los valores exactos de un atributo privado. El ámbito de instancia es, por mucho, el tipo más común de ámbito.

# Resumen

Para completar sus nociones de clases y la forma en que se conectan, es necesario comprender algunas relaciones adicionales. Una agregación establece una asociación para conformar un todo: una clase "todo" se genera de clases que la componen. Un componente en una agregación puede ser parte de más de un todo. Una composición es una conformación muy íntimamente ligada con la agregación en el sentido de que un componente de una composición puede ser parte solamente de un todo. La representación del UML de las agregaciones es similar a la representación de las composiciones. La línea de asociación que une la parte con un todo tiene un rombo. En una agregación, el rombo no está relleno, en tanto que en una composición sí lo está.

Un diagrama de contexto enfoca la atención en una clase específica dentro de un sistema. Un diagrama de contexto de composición es como un mapa detallado de un mapa mayor. Muestra un diagrama de clases anidado dentro de un gran símbolo rectangular de clase. Un diagrama de contexto de sistema muestra la forma en que el diagrama de clases compuestas se relaciona con otros objetos del sistema.

Una realización es una asociación entre una clase y una interfaz, una colección de operaciones que cierta cantidad de clases podrá utilizar. Una interfaz se representa como una clase sin atributos. Para distinguirla de una clase cuyos atributos hayan sido omitidos del diagrama, el estereotipo «interfaz» aparecerá por encima del nombre de la interfaz. Otra posibilidad es la de anteceder el nombre de la interfaz con una "I" mayúscula. La realización se representa en el UML mediante una línea discontinua con una punta de flecha en forma de triángulo sin rellenar que conecta a la clase con la interfaz. Otra forma para representar una realización es con una línea continua que conecte a una clase con un pequeño círculo, para que el círculo se interprete como la interfaz.

www.elsolucionario.net

www.elsolucionario.net

En términos de visibilidad, todas las operaciones en una interfaz son *públicas*, de modo que cualquier clase podrá utilizarlas. Los otros dos niveles de visibilidad son *protegido* (la funcionalidad se extiende a las clases secundarias de aquella que contiene los atributos y operaciones) y *privado* (atributos y operaciones que se pueden utilizar sólo dentro de la clase que los contiene). Un signo de suma (+) denota a la visibilidad pública, el símbolo de número (#) la protegida y el guión (-) la privada.

El ámbito es otro aspecto de los atributos y operaciones. En un ámbito de instancia, cada objeto de una clase cuenta con su propio valor en un atributo u operación. En un ámbito de archivador, sólo hay un valor para un atributo u operación en particular a través de un conjunto de objetos de una clase. Los objetos que no estén en este conjunto no podrán acceder al valor contenido en el ámbito de archivador.

# Preguntas y respuestas

**P ¿Se considera transitiva a la agregación? Es decir, si la clase 3 es un componente de la clase 2, y la clase 2 es un componente de la clase 1, ¿la clase 3 será un componente de la clase 1?**

**R** Así es, la agregación es transitiva. En nuestro ejemplo, los botones y la bola del ratón son parte del ratón, a la vez que son parte de la computadora.

**P ¿La palabra "interfaz" implica "interfaz de usuario" o GUI?**

**R** No. Es algo más genérico. Una interfaz es tan sólo un conjunto de operaciones que una clase presenta a las demás clases. De hecho, una de estas operaciones podría ser (aunque no necesariamente) la del usuario.

# Taller

El cuestionario y los ejercicios verificarán y fortalecerán su conocimiento respecto al tema de las agregaciones, composiciones, contextos e interfaces. Las respuestas las podrá ver en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿Cuál es la diferencia entre una agregación y una composición?
2. ¿Qué es la realización?
3. Mencione los tres niveles de visibilidad y describa lo que significa cada uno de ellos.

## Ejercicios

1. Cree un diagrama de contexto de composición de una revista. Tome en cuenta la tabla de contenido, la editorial, los artículos y las columnas. Luego, cree un diagrama de contexto del sistema que muestre a la revista junto con el suscriptor y el comprador en el puesto de revistas.

www.elsolucionario.net

www.elsolucionario.net

2. En la actualidad, el tipo más popular de GUI es la interfaz WIMP (ventanas, iconos, menús y puntero, por sus siglas en inglés). Dibuje un diagrama de clases de la interfaz WIMP, y haga uso de todo el conocimiento adecuado del UML que ha adquirido hasta ahora. Además de las clases indicadas en las siglas, incluya los elementos relacionados como las barras de desplazamiento y el cursor, así como cualquiera de las otras clases necesarias.

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 6

# Introducción a los casos de uso

Ahora que ha visto lo correspondiente a las clases y sus relaciones, es el momento de volver nuestra atención a otra área principal del UML: los casos de uso.

En esta hora se tratarán los siguientes temas:

- Qué son los casos de uso
- Importancia de los casos de uso
- Inclusión de los casos de uso
- Extensión de los casos de uso
- Inicio de un análisis de un caso de uso

En las tres horas anteriores hemos visto los diagramas que proporcionan una idea estática de las clases en un sistema. Ahora veremos a los diagramas que establecen una idea dinámica y mostraremos la forma en que el sistema y sus clases cambian con el tiempo. Las ideas estáticas ayudan a que un analista se comunique con un cliente. La idea dinámica, como verá, ayudará al analista a comunicarse con un grupo de desarrolladores, y ayudará a estos últimos a crear programas.

www.elsolucionario.net

www.elsolucionario.net

El cliente y el equipo de desarrollo conforman un importante conjunto de integrantes en un sistema. No obstante, una parte de igual importancia no se ha tomado en cuenta: el usuario. Ni la idea estática ni la dinámica mostrarán el comportamiento del sistema desde el punto de vista del usuario. Comprender tal punto de vista es clave para generar sistemas que sean tanto útiles como funcionales; esto es, que cumplan con los requerimientos y que sea fácil (e, incluso, divertido) trabajar con ellos.

El modelado de un sistema desde el punto de vista de un usuario es el trabajo de los casos de uso. En esta hora comprenderá todo lo relacionado con los casos de uso y su función. En la siguiente hora aprenderá a utilizar el diagrama de casos de uso del UML para visualizar un caso de uso.

# Qué son los casos de uso

Recientemente adquirí una máquina de fax. Cuando fui a comprarla, en un almacén de venta de equipo para oficinas, encontré una enorme gama de opciones. ¿Cómo hice para decidirme por una en particular? Me pregunté exactamente qué es lo que deseaba hacer con una máquina de fax. ¿Qué características deseaba? ¿Cuáles funciones necesitaba que tuviera? ¿Deseaba utilizar papel común o térmico? ¿Quería generar copias? ¿Conectarlo a mi computadora? ¿Utilizarlo como digitalizador? ¿Tendría que enviar faxes a tal velocidad que necesitaría una función de marcado rápido? ¿Querría utilizar la máquina de fax para diferenciar entre una llamada telefónica y un fax entrante?

Todos seguimos un procedimiento como éste cuando realizamos una compra que no sea impulsiva. Lo que hacemos es seguir un tipo de *análisis del caso de uso*: nos preguntamos cómo utilizaremos el producto o sistema que queremos comprar, de modo que podamos obtener algo que cumpla con nuestras necesidades. Lo importante es saber cuáles son esos requerimientos.

Este tipo de análisis es particularmente crucial para la fase de análisis del desarrollo de un sistema. La forma en que los usuarios utilicen un sistema le da la pauta para lo que diseñará y creará.

El caso de uso es una estructura que ayuda a los analistas a trabajar con los usuarios para determinar la forma en que se usará un sistema. Con una colección de casos de uso se puede hacer el bosquejo de un sistema en términos de lo que los usuarios intenten hacer con él.

**TERMINO NUEVO** Imagínese al caso de uso como una colección de situaciones respecto al uso de un sistema. Cada escenario describe una secuencia de eventos. Cada secuencia se inicia por una persona, otro sistema, una parte del hardware o por el paso del tiempo. A las entidades que inician secuencias se les conoce como *actores*. El resultado de la secuencia debe ser algo utilizable ya sea por el actor que la inició, o por otro actor.

www.elsolucionario.net

www.elsolucionario.net

# Importancia de los casos de uso

Así como el diagrama de clases es un buen medio para estimular a un cliente a que hable respecto a un sistema desde su propio punto de vista, el caso de uso es una excelente herramienta para estimular a que los usuarios potenciales hablen, de un sistema, desde sus propios puntos de vista. No siempre es fácil para los usuarios explicar cómo pretenden utilizar un sistema. Puesto que el desarrollo tradicional de los sistemas era, con frecuencia, algo así como una ciencia oculta, con muy poca información para los usuarios, a aquellos que osaban preguntar se les daba información muy poco explícita o ciertamente confusa respecto a lo que utilizarían.

La idea es involucrar a los usuarios en las etapas iniciales del análisis y diseño del sistema. Esto aumenta la probabilidad de que el sistema sea de mayor provecho para la gente a la que supuestamente ayudará, en lugar de ser un manojo de expresiones de computación incomprensibles e inmanejables por los usuarios finales.

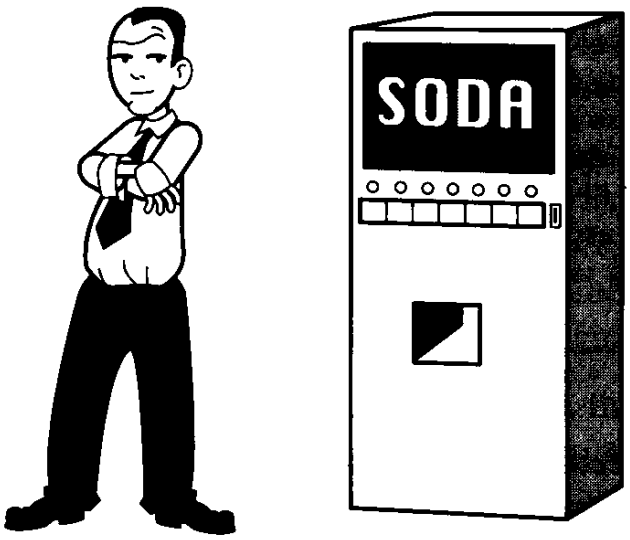
# Un ejemplo: la máquina de gaseosas

Suponga que empezará a diseñar una máquina despachadora de gaseosas. Para obtener el punto de vista del interesado, entrevistará a varios usuarios potenciales respecto a la manera en que utilizarán dicha máquina.

Dado que la función principal de una máquina de gaseosas es permitir a un cliente adquirir una lata de gaseosa, probablemente las personas le dirán que se enfrentará a diversos escenarios —un caso de uso, en otras palabras— que podría etiquetar como "Comprar gaseosa". Examinemos cada posible escenario en este caso de uso. Recuerde que tales escenarios podrían aparecer durante la conversación con los usuarios.

**FIGURA 6.1**

*Un caso de uso establece un conjunto de escenarios para realizar algo útil para un actor. En este ejemplo, un caso de uso es "Comprar gaseosa".*




www.elsolucionario.net

www.elsolucionario.net

## El caso de uso "Comprar gaseosa"

El actor, en este caso de uso, es un cliente que desea comprar una lata de gaseosa. El escenario iniciará cuando el cliente inserte dinero, posteriormente realizará una selección; y si todo funciona bien, la máquina contará con, al menos, una lata de la gaseosa elegida, misma que pondrá al alcance del cliente.

Además de la secuencia, hay otros aspectos del escenario anterior que merecen cierta consideración. ¿Qué condiciones llevaron al cliente a iniciar el escenario en el caso de uso "Comprar gaseosa"? La sed es la más obvia. ¿Qué se obtiene como resultado de tal escenario? Nuevamente, lo obvio es que el cliente tenga una gaseosa en su poder.

¿Lo que he descrito es la única posibilidad de "Comprar gaseosa"? Habría otras cuestiones que saltarían a la vista; por ejemplo, es posible que la máquina no tenga la gaseosa que desee el cliente; también es posible que el cliente no tenga el importe exacto de la gaseosa. ¿Cómo diseñaría a la máquina de gaseosas para controlar tales escenarios?

Veamos el caso en que la máquina se haya quedado sin gaseosa, otra secuencia de pasos en el caso de uso "Comprar gaseosa". Imagínelo como una ruta alternativa dentro del caso de uso. El cliente inicia el caso de uso al insertar dinero en la máquina y posteriormente hace una selección. La máquina no cuenta con ninguna lata de la gaseosa seleccionada, por lo que mostrará un mensaje al cliente que indicará que no tiene de esa marca. Lo ideal sería que el mensaje le pida al cliente que haga otra selección. La máquina también debería dar la opción de devolver el dinero al cliente. En este punto, el cliente selecciona otra marca que la maquina entregará (siempre y cuando cuente con provisiones de esta marca), o devolverá el dinero. La condición previa es un cliente sediento y el resultado es una lata de gaseosa o la devolución del dinero.

Claro que el escenario de quedarse sin gaseosa sería posible: el mensaje "No hay de esta marca" podría aparecer en cuanto las provisiones de la máquina se acabaran y permanecer a la vista hasta que la máquina sea reabastecida. En tal caso, el usuario podría no insertar el dinero en primera instancia. El cliente para el que usted diseñará la máquina podría preferir el primer escenario: si el cliente ya insertó dinero, la tendencia podría ser hacer otra selección en lugar de pedir a la máquina que lo devuelva.

Analicemos ahora el escenario de la cantidad de dinero incorrecta. Nuevamente, el usuario inicia el caso de uso en la forma usual y posteriormente hace una selección. Asumamos que la máquina tiene provisión de la marca elegida. En la máquina hay una reserva de moneda fraccionaria y devuelve la diferencia al despachar la gaseosa. Si la

www.elsolucionario.net

www.elsolucionario.net

máquina no cuenta con una reserva de moneda fraccionaria, devolverá el dinero y mostrará un mensaje que pida al usuario el importe exacto. La condición previa es la ya indicada. El resultado será una lata de gaseosa junto con el cambio, o la devolución del dinero originalmente depositado.

Otra posibilidad es que tan pronto como se agote la moneda fraccionaria, aparezca un mensaje que informe a los clientes que se requiere el importe exacto. El mensaje permanecería a la vista hasta que la máquina sea reabastecida con moneda fraccionaria.

## Casos de uso adicionales

Ya ha examinado a la máquina de gaseosas desde el punto de vista de un usuario: el cliente. Hay otros usuarios que intervienen, como el proveedor que tiene que reabastecer a la máquina, el recolector de dinero (que tal vez sea el mismo que el proveedor) que tiene que recoger el dinero acumulado en la alcancía de la máquina, etcétera. Esto nos indica que debemos crear al menos dos casos de uso: "Reabastecer" y "Recolectar dinero", cuyos detalles surgirán durante las entrevistas con los proveedores y los recolectores.

Veamos el caso de uso de "Reabastecer". El proveedor inicia este caso de uso dado que algún intervalo (digamos, dos semanas) ha pasado. El representante del proveedor le quita el seguro a la máquina (tal vez mediante una llave y un cerrojo, pero eso entra dentro de la implementación), jala la puerta para abrir la máquina, y llena el compartimiento de cada marca hasta su capacidad. El representante también rellena la reserva de moneda fraccionaria. Luego, cierra el frente de la máquina y vuelve a poner el seguro. La condición previa es el paso del intervalo, el resultado es que el proveedor cuenta con un nuevo conjunto de ventas potenciales.

Para el caso de uso de "Recolectar el dinero", el recolector inicia debido también a que ha pasado cierto tiempo. La persona deberá seguir la misma secuencia que en "Reabastecer" para abrir la máquina. El recolector sacará el dinero de la máquina y seguirá los pasos de "Reabastecer" para cerrar y poner el seguro a la máquina. La condición previa es el paso del intervalo y el resultado es el dinero en las manos del recolector.

Vea que cuando derivamos un caso de uso, no nos preocupamos por la forma de implementarlo. En nuestro ejemplo, no nos interesamos en los aspectos internos de la máquina de gaseosas. Tampoco por la forma en que funcione el mecanismo de refrigeración, o por la forma en que la máquina controle la cantidad de dinero que contenga. Tan sólo intentamos ver la forma en que la máquina lucirá para alguien que tenga que utilizarla.

El objetivo es derivar una colección de casos de uso que, finalmente, mostraremos a las personas que diseñen la máquina de gaseosas y a las personas que la construirán. Por añadidura, nuestros casos de uso reflejan lo que los clientes, recolectores y proveedores desean, por lo que el resultado será una máquina que todos esos grupos puedan utilizar con facilidad.

www.elsolucionario.net

www.elsolucionario.net

# Inclusión de los casos de uso

En los casos de uso "Reabastecer" y "Recolectar dinero", tal vez distinguió ciertos pasos en común. Ambos empezaban con abrir la máquina, y finalizaban con el cierre de la máquina y su aseguramiento. ¿Podríamos eliminar la duplicación de pasos de un caso de uso al otro?

Sí podemos. La forma de hacerlo es tomar cada secuencia de pasos en común y conformar un caso de uso adicional a partir de ellos. Combinemos los pasos necesarios para "quitar el seguro" y "abrir la máquina" y llamémoslos "Exhibir el interior" y los pasos "cerrar la máquina" y "asegurarla" en otro caso de uso llamado "Cubrir el interior".

Con estos nuevos casos de uso a la mano, el caso de uso "Reabastecer" iniciaría con el caso de uso "Exhibir el interior". Luego, el representante del proveedor seguiría los pasos ya indicados, y concluiría con el caso de uso "Cubrir el interior". De forma similar, el caso de uso "Recolectar dinero" iniciaría con "Exhibir el interior", procedería como se indicó, y finalizaría con el caso de uso "Cubrir el interior".

Como ve, "Reabastecer" y "Recolectar dinero" incluyen los nuevos casos de uso. Por ello, a esta técnica de aprovechamiento de un caso de uso se le conoce como *inclusión de un caso de uso.*

La inclusión de un caso de uso también se conoce como *usar un caso de uso.* Creo que el término *incluir* tiene dos ventajas. La primera, es más clara: los pasos en un caso de uso, incluyen los de otro. La segunda, se evita la confusión potencial de las palabras "usar" y "uso" en un contexto tan estrecho. Así, no tendremos que decir "promover el uso mediante el uso reiterativo de un caso de uso".

www.elsolucionario.net

# Extensión de los casos de uso

Es posible volver a utilizar un caso de uso de una forma distinta a una inclusión. En ocasiones crearemos un caso de uso agregándole algunos pasos a un caso de uso existente.

Regresemos al caso de uso "Reabastecer". Antes de colocar nuevas latas de gaseosas en la máquina, suponga que el representante del proveedor nota las marcas que se han vendido bien, así como las que no se han vendido tan bien. En lugar de sólo reabastecer todas las marcas, el representante podría sacar aquellas que no se han vendido bien, y reemplazarlas por latas de las marcas que han probado ser más populares. De esta forma, tendría que indicar al frente de la máquina el nuevo surtido de marcas disponibles.

Si agregamos estos pasos a "Reabastecer", tendremos un nuevo caso de uso que llamaríamos "Reabastecer de acuerdo a las ventas". Este nuevo caso de uso es una extensión del original, acción a la que se le conoce como *extensión de un caso de uso.*

www.elsolucionario.net

# Inicio del análisis de un caso de uso

En nuestro caso, nos hemos involucrado directamente con los casos de uso y nos hemos enfocado en algunos de ellos. En el mundo real, por lo general, seguirá un conjunto de procedimientos cuando empiece un análisis de casos de uso.

Empezará con entrevistas a los clientes (y entrevistas con expertos) que lo lleven a los diagramas iniciales de clases que indicamos en la hora 3. Esto le dará cierta idea del área en la que trabajará y una familiaridad con los términos que utilizará. Posteriormente, contará con un fundamento para hablar con los usuarios.

Entrevistará a los usuarios (preferentemente en grupos) y les pedirá que le indiquen todo lo que ellos harían con el sistema que usted diseñará. Sus respuestas conformarán un conjunto candidato de casos de uso. Luego, es importante describir brevemente cada caso de uso. También tendrá que derivar una lista de todos los actores que iniciarán y se beneficiarán de los casos de uso. Cuenta más información obtenga en esta fase, aumentará su aptitud para hablar con los usuarios en su propio idioma.

Los casos de uso aparecerán en varias fases del proceso de desarrollo. Le ayudarán con el diseño de una interfaz del usuario, coadyuvarán con las opciones de desarrollo de los programadores y establecerán las bases para probar el sistema recién generado.

Para mayor información en el tema del análisis de los casos de uso, va a tener que aplicar el UML, y ello se hará en la siguiente hora.

# Resumen

El caso de uso es una estructura para describir la forma en que un sistema lucirá para los usuarios potenciales. Es una colección de escenarios iniciados por una entidad llamada actor (una persona, un componente de hardware, un lapso u otro sistema). Un caso de uso debería dar por resultado algo de valor ya sea para el actor que lo inició o para otro.

Es posible volver a utilizar casos de uso. Una forma ("inclusión") es utilizar los pasos de un caso de uso como parte de la secuencia de pasos de otro caso de uso. Otra forma ("extensión") es crear un nuevo caso de uso mediante la adición de pasos a un caso de uso existente.

La entrevista directa con los usuarios es la mejor técnica para derivar casos de uso. Cuando se deriva un caso de uso, es importante destacar las condiciones para iniciar el caso de uso, y los resultados obtenidos como consecuencia del mismo.

Hará las entrevistas a los usuarios después de entrevistar a los clientes y generar una lista de prospectos de clases. Esto le dará un fundamento en la terminología que utilizará para hablar con los usuarios. Es una buena idea entrevistar a un grupo de usuarios. El objetivo es derivar un conjunto candidato de casos de uso y todos los posibles actores.

www.elsolucionario.net

www.elsolucionario.net

# Preguntas y respuestas

**P En realidad ¿para qué necesito el concepto del caso de uso? ¿Qué no sólo podríamos preguntar a los usuarios lo que deseen ver en el sistema y dejarlo así?**

**R** En realidad, no. Tenemos que crear una estructura de lo que los usuarios nos digan, y los casos de uso la proporcionan. La estructura se vuelve útil cuando tiene que llevar los resultados de sus entrevistas con los usuarios y comunicarlos a los clientes y desarrolladores.

**P ¿Qué tan difícil es derivar los casos de uso?**

**R** De acuerdo con mi experiencia, el listado de casos de uso —al menos los de alto nivel— no es muy complejo. Hay ciertas dificultades al profundizar en cada una e intentar lograr que los usuarios listen los pasos de cada escenario. Cuando genere un sistema que reemplace una manera existente de hacer las cosas, los usuarios típicamente ya sabrán los pasos bastante bien y los habrán utilizado con tanta regularidad que se les dificultará estructurarlos. Es una buena idea tener un panel de usuarios, ya que la discusión en grupo por lo general trae consigo ideas que un usuario en particular podría tener problemas para expresar.

# Taller

Esta hora se basó en teoría más que en el UML. En este taller, el objetivo será comprender los conceptos teóricos y aplicarlos en diversos contextos. La práctica, que veremos en la siguiente hora, le ayudará a reafirmar los conceptos cuando aprenda a visualizarlos mediante el UML. Las respuestas aparecen en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿Cómo se llama a la entidad que inicia un caso de uso?
2. ¿Qué se entiende con "incluir un caso de uso"?
3. ¿Qué se entiende con "extender un caso de uso"?
4. ¿Un caso de uso es lo mismo que un escenario?

## Ejercicios

1. En el caso del ejemplo de la máquina de gaseosas, cree otro caso de uso que incluya a los casos de uso "Exhibir el interior" y "Cubrir el interior".
2. Los casos de uso pueden ayudarle a analizar un negocio y un sistema. Imagine a una gran tienda de equipos de cómputo que venda hardware, periféricos y software. ¿Quiénes serían los actores? ¿Cuáles serían algunos de los principales casos de uso? ¿Cuáles serían algunos de los escenarios dentro de cada caso de uso?

www.elsolucionario.net

www.elsolucionario.net

# HORA 7

# Diagramas de casos de uso

El caso de uso es un poderoso concepto que ayuda a un analista a comprender la forma en que un sistema deberá comportarse. Le ayuda a obtener los requerimientos desde el punto de vista del usuario. Es necesario aprender a visualizar los conceptos del caso de uso que conoció en la hora anterior.

En esta hora se tratarán los siguientes temas:

- Representación de un modelo de caso de uso
- Concepción de las relaciones entre casos de uso
- Diagramas de casos de uso en el proceso de análisis
- Aplicación de los modelos de caso de uso
- Verá la idea general del UML

El caso de uso es muy poderoso, pero lo es aún más cuando se visualiza por medio del UML. Esta visualización le permitirá mostrar los casos de uso a los usuarios para que ellos le puedan dar mayor información. Es un hecho que los usuarios con frecuencia saben más de lo que dicen: el caso de uso ayuda a romper el hielo. A su vez, una representación visual le ayuda a combinar los diagramas de casos de uso con otro tipo de diagramas.

Una de las finalidades del proceso de análisis de un sistema es generar una colección de casos de uso. La idea es tener la posibilidad de catalogar y

www.elsolucionario.net

www.elsolucionario.net

hacer referencia a esta colección, que sirve como el punto de vista de los usuarios acerca del sistema. Cuando llegue el momento de actualizar el sistema, el catálogo de casos de uso funcionará como un fundamento para obtener los requerimientos de la actualización.

# Representación de un modelo de caso de uso

Hay un actor que inicia un caso de uso y otro (posiblemente el que inició, pero no necesariamente) que recibirá algo de valor de él. La representación gráfica es directa. Una elipse representa a un caso de uso, una figura agregada representa a un actor. El actor que inicia se encuentra a la izquierda del caso de uso, y el que recibe a la derecha. El nombre del actor aparece justo debajo de él, y el nombre del caso de uso aparece ya sea dentro de la elipse o justo debajo de ella. Una línea asociativa conecta a un actor con el caso de uso, y representa la comunicación entre el actor y el caso de uso. La línea asociativa es sólida, como la que conecta a las clases asociadas.

Uno de los beneficios del análisis del caso de uso es que le muestra los confines entre el sistema y el mundo exterior. Generalmente, los actores están fuera del sistema, mientras que los casos de uso están dentro de él. Utilizará un rectángulo (con el nombre del sistema en algún lugar dentro de él) para representar el confín del sistema. El rectángulo envuelve a los casos de uso del sistema.

**TÉRMINO NUEVO** Los actores, casos de uso y líneas de interconexión componen un modelo de caso de uso. La figura 7.1 le muestra estos símbolos.

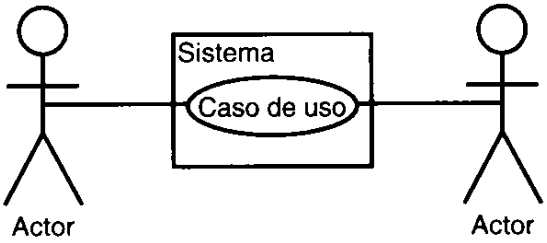



**FIGURA 7.1**

*En un modelo de caso de uso, una figura agregada representa a un actor, una elipse a un caso de uso y una línea asociativa representa la comunicación entre el actor y el caso de uso.*

## Una nueva visita a la máquina de gaseosas

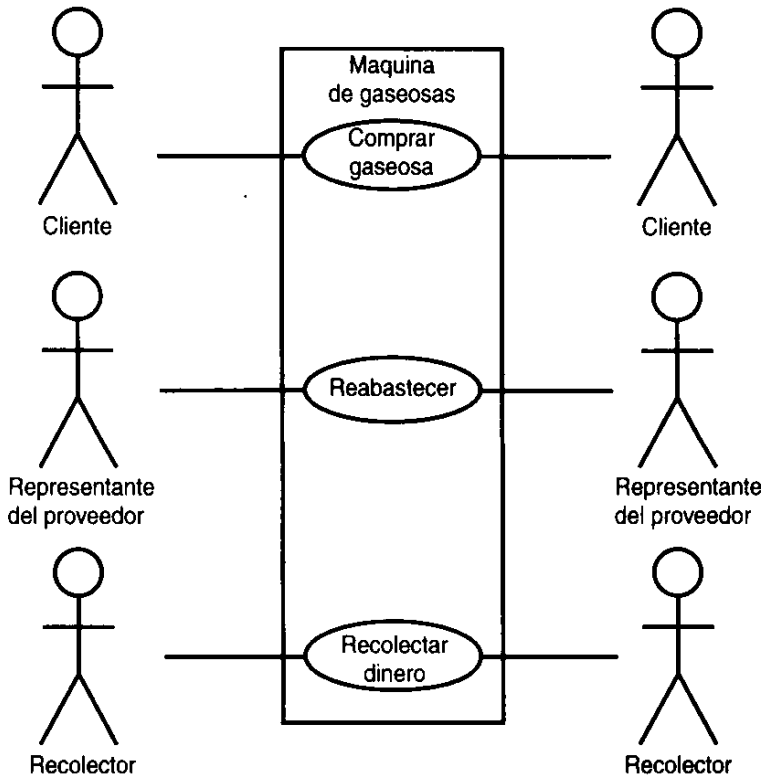
Apliquemos los símbolos al ejemplo de la hora anterior. Como recuerda, desarrolló los casos de uso para una máquina de gaseosas. El caso de uso "Comprar gaseosa" se encuentra dentro del sistema junto con "Reabastecer" y "Recolectar dinero". Los actores son el Cliente, Representante del proveedor y el Recolector. La figura 7.2 le muestra un modelo UML de caso de uso para una máquina de gaseosas.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 7.2**

*Un modelo de caso de uso proveniente de la máquina de gaseosas de la hora 6.*




## Secuencia de pasos en los escenarios

Cada caso de uso es una colección de escenarios y cada escenario es una secuencia de pasos. Como puede ver, tales pasos no aparecen en el diagrama. No se encuentran en notas adjuntas a los casos de uso. Aunque el UML no lo prohíbe, la claridad es clave en la generación de cualquier diagrama y el adjuntar notas a cada caso de uso podría volverlo confuso. ¿Cómo y dónde haría la secuencia de pasos?

El uso de los diagramas de casos de uso será, por lo general, parte de un documento de diseño que el cliente y el equipo de diseño tomarán como referencia. Cada diagrama tendrá su propia página, de igual manera, cada escenario de caso de uso tendrá su propia página, donde se listará en modo de texto a:

El actor que inicia al caso de uso

Condiciones previas para el caso de uso

Pasos en el escenario

Condiciones posteriores cuando se finaliza el escenario

El actor que se beneficia del caso de uso

También puede enumerar las conjeturas del escenario (por ejemplo, que un cliente a la vez utilizará la máquina de gaseosas) y una breve descripción de una sola frase del escenario.

www.elsolucionario.net

www.elsolucionario.net

La hora 6, "Introducción a los casos de uso", presentó algunos escenarios alternativos del caso de uso "Comprar gaseosa". En su descripción, también podría poner estos escenarios de manera separada ("Sin el producto" y "Cambio incorrecto"), o podría considerarlos como excepciones al primer escenario del caso de uso. La forma exacta de hacerlo sólo le concernirá a usted, su cliente y los usuarios.

Para mostrar los pasos en un escenario, hay otra posibilidad que es utilizar un diagrama de actividades UML sobre el cual hablaremos en la hora 11, "Diagramas de actividades".

# Concepción de las relaciones entre casos de uso

TERMINO NUEVO El ejemplo de la hora 6 también mostró dos formas en que los casos de uso se pueden relacionar entre sí. Una de ellas, la *inclusión*, le permite volver a utilizar los pasos de un caso de uso dentro de otro. La otra, *extensión*, le permite crear un caso de uso mediante la adición de pasos a uno existente.

TERMINO NUEVO Existen otros dos tipos de relaciones que son generalización y agrupamiento. Como en las clases, la *generalización* cuenta con un caso de uso que se hereda de otro. El *agrupamiento* es una manera sencilla de organizar los casos de uso.

## Inclusión

Examinemos los casos de uso "Reabastecer" y "Recolectar dinero" del ejemplo de la hora 6. Ambos se inician mediante la apertura de la máquina, y finalizan con el cierre y sellado de la misma. El caso de uso "Exhibir el interior" se creó para capturar el primer par de pasos; y "Cubrir el interior" para el segundo. Tanto "Reabastecer", como "Recolectar dinero" incluyen este par de casos de uso.

Para representar a la inclusión utilizará el símbolo que usó para la dependencia entre clases: una línea discontinua con una punta de flecha que conecta las clases apuntando hacia la clase dependiente. Justo sobre la línea, agregará un estereotipo: la palabra "incluir" bordeada por dos pares de paréntesis angulares. La figura 7.3 le muestra la relación de «inclusión» en el modelo de caso de uso de la máquina de gaseosas.

Tenga en cuenta que un caso de uso incluido nunca aparecerá solo. Simplemente funciona como parte de un caso de uso que lo incluya.

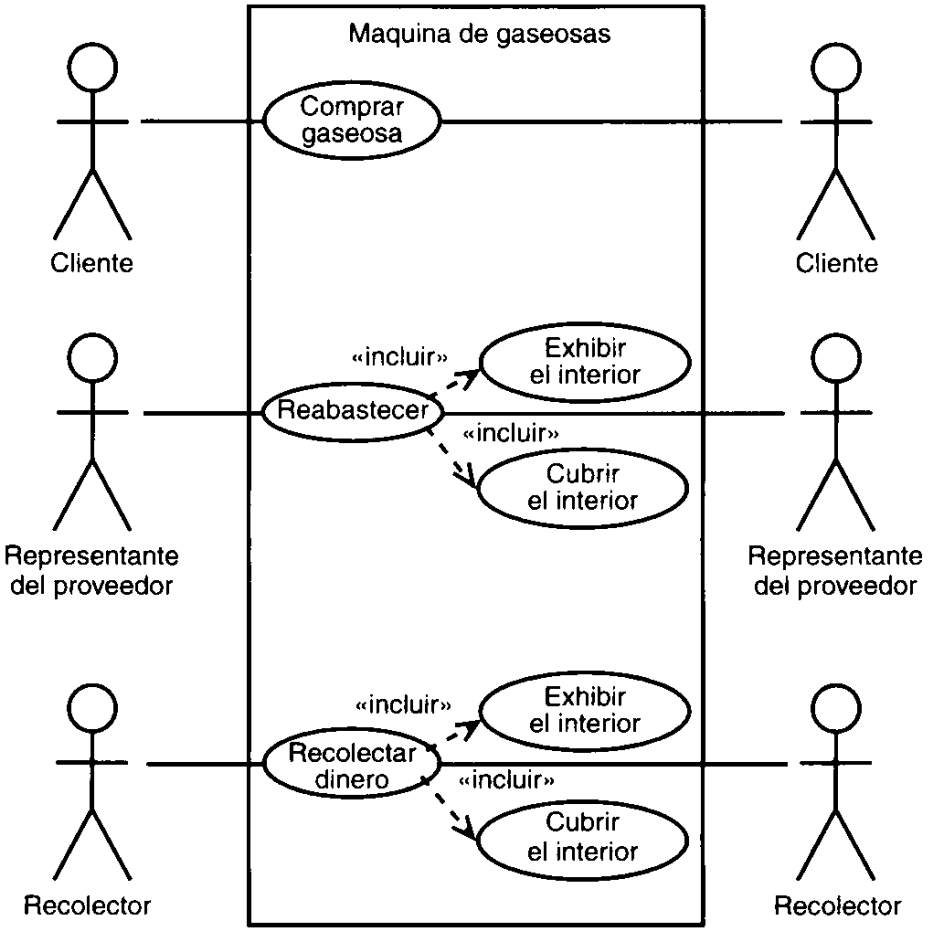
En la notación de texto que sigue los pasos en la secuencia, indicará los casos de uso incluidos. El primer paso en el caso de uso "Reabastecer" podría ser «incluir» (Exhibir el interior).

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 7.3**

*El modelo de caso de uso en la máquina de gaseosas con la inclusión.*




## Extensión

**TÉRMINO NUEVO** La hora 6 mostró que el caso de uso "Reabastecer" podría ser la base de otro caso de uso: "Reabastecer de acuerdo a las ventas". En lugar de sólo reabastecer la máquina de gaseosas para que todas las marcas tengan la misma cantidad de latas, el representante podría anotar aquellas que se venden mejor y reabastecer acorde con ello. Por lo que podemos decir que el nuevo caso de uso *extiende* al original dado que agrega otros pasos a la secuencia del caso de uso original, que se conoce como el caso de uso *base*.

**TÉRMINO NUEVO** La extensión sólo se puede realizar en puntos indicados de manera específica dentro de la secuencia del caso de uso base. A estos puntos se les conoce como *puntos de extensión*. En el caso de uso Reabastecer, los nuevos pasos (anotar las ventas y abastecer de manera acorde) se darían luego que el representante haya abierto la máquina y esté listo para llenar los compartimientos de las marcas de gaseosas. En este ejemplo, el punto de extensión es "Llenar los compartimientos".

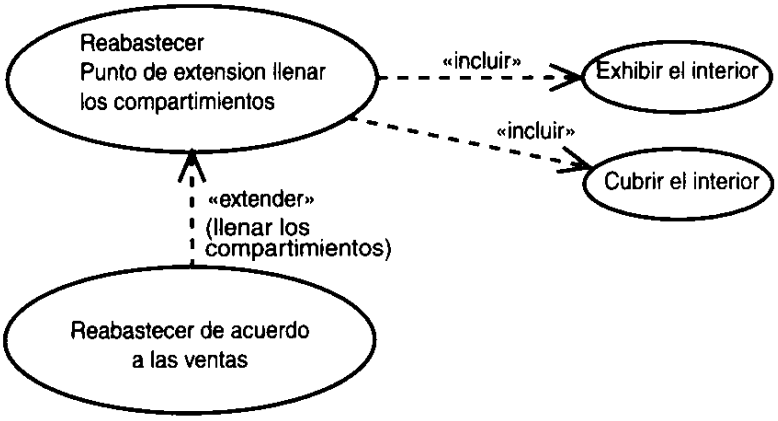
Como la inclusión, podrá concebir la extensión con una línea de dependencia (línea discontinua con punta una punta de flecha), junto con un estereotipo que muestra "extender" entre paréntesis angulares. Dentro del caso de uso básico, el punto de extensión aparecerá debajo del nombre del caso de uso. La figura 7.4 le muestra la relación de extensión para "Reabastecer" y "Reabastecer de acuerdo a las ventas", junto con la inclusión de relaciones para "Reabastecer" y "Recolectar dinero".

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 7.4**

*Un diagrama de casos de uso que muestra la extensión y la inclusión.*



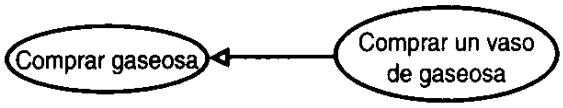

## Generalización

Las clases pueden heredarse entre sí y esto también se aplica a los casos de uso. En la herencia de los casos de uso, el caso de uso secundario hereda las acciones y significado del primario, y además agrega sus propias acciones. Puede aplicar el caso de uso secundario en cualquier lugar donde aplique el primario.

En el ejemplo, deberá imaginar un caso de uso "Comprar un vaso de gaseosa" que se hereda de "Comprar gaseosa". El caso de uso secundario tiene acciones como "agregar hielo" y "mezclar marcas de gaseosas". Modelará la generalización de casos de uso de la misma forma que lo hace con las clases: con líneas continuas y una punta de flecha en forma de triángulo sin rellenar que apunta hacia el caso de uso primario, como se muestra en la figura 7.5.

**FIGURA 7.5**

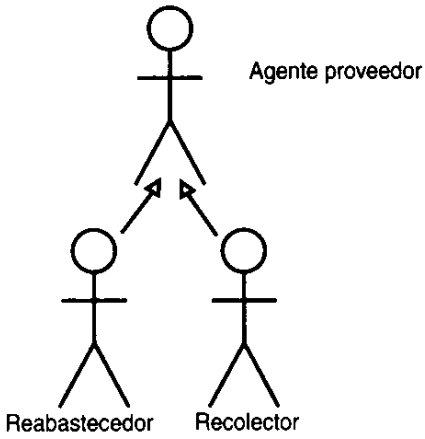
*Un caso de uso puede heredar el sentido y comportamiento de otro.*




La relación de generalización puede establecerse entre actores, así como entre casos de uso. Quizá tenga personificados al representante del proveedor, al recolector y al agente del proveedor. Si cambia el nombre del representante como Reabastecedor, tanto éste como el Recolector serán secundarios del Agente Proveedor, como muestra la figura 7.6.

**FIGURA 7.6**

*Como las clases y los casos de uso, los actores pueden estar en una relación de generalización.*




www.elsolucionario.net

www.elsolucionario.net

## Agrupamiento

En ciertos diagramas de casos de uso, podría tener varios casos de uso que querrá organizar. Esto puede ocurrir cuando un sistema consta de varios subsistemas. Otra posibilidad sería cuando entrevista a los usuarios para obtener los requerimientos de un sistema. Cada requerimiento podría ser representado como un caso de uso por separado. Necesitará alguna forma de ordenar los requerimientos por categorías.

La forma más directa de organizar sería agrupar en un paquete los casos de uso que se relacionen. Recuerde que un paquete aparece como una carpeta tabular. Los casos de uso agrupados aparecerán dentro de la carpeta.

# Diagramas de casos de uso en el proceso de análisis

Con el ejemplo dado y con el cual ha trabajado, aplicó directamente la simbología del caso de uso. Ahora nos regresaremos un poco y colocaremos los casos de uso en el contexto de un esfuerzo de análisis.

Las entrevistas al cliente deberán iniciar el proceso. Estas entrevistas producirán diagramas de clases que fungirán como las bases de su conocimiento para el dominio del sistema (el área en el cual resolverá los problemas). Una vez que conozca la terminología general del área del cliente, estará listo para hablar con los usuarios.

Las entrevistas con los usuarios comienzan en la terminología del dominio, aunque deberán alternarse hacia la terminología de los usuarios. Los resultados iniciales de las entrevistas deberán revelar a los actores y casos de uso de alto nivel que describirán los requerimientos funcionales en términos generales. Esta información establece los confines y ámbito del sistema.

Las entrevistas posteriores con los usuarios profundizarán en estos requerimientos, lo que dará por resultado modelos de casos de uso que mostrarán los escenarios y las secuencias detalladamente. Esto podría resultar en otros casos de uso que satisfagan las relaciones de inclusión y extensión. En esta fase, es importante confiar en su comprensión del dominio (a partir de los diagramas de clases derivados de las entrevistas con el cliente). Si no comprende adecuadamente el dominio, podría crear demasiados casos de uso y demasiados detalles (situación que podría, definitivamente, obstaculizar el diseño y el desarrollo).

# Aplicación de los modelos de caso de uso

Para ayudarle a comprender con más profundidad los modelos de casos de uso y cómo aplicarlos, vamos a ver un ejemplo más complejo que una máquina de gaseosas. Suponga que deberá diseñar una red de área local (LAN) para una firma de consultoría, y que tendrá que comprender la funcionalidad para poder crearla. ¿Cómo empezaría?

www.elsolucionario.net

www.elsolucionario.net

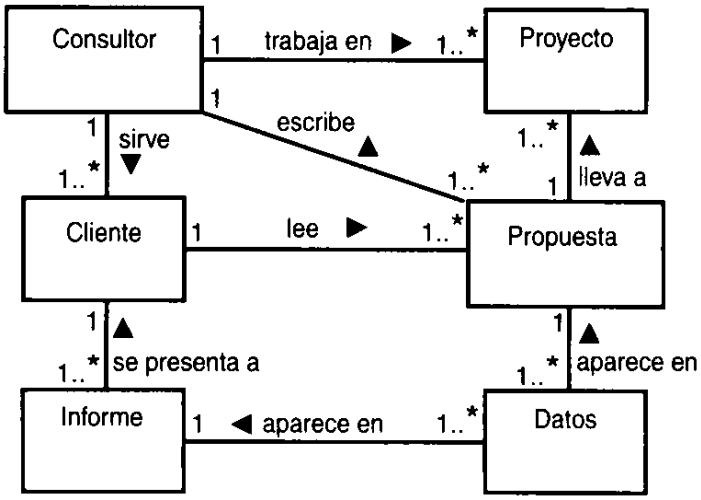
Una LAN es una red de comunicaciones que una organización utiliza en un ámbito limitado. Permite a los usuarios compartir recursos e información.

## Comprensión del dominio

Empecemos con las entrevistas al cliente para crear un diagrama de clases que refleje cómo es la vida en el mundo de la consultoría. El diagrama de clases podría incluir las siguientes clases: Consultor, Cliente, Proyecto, Propuesta, Datos e Informe. La figura 7.7 le muestra la forma en que podría lucir el diagrama.

**FIGURA 7.7**

*Un diagrama de clases para el mundo de la consultoría.*




## Comprensión de los usuarios

Ahora que el dominio está a la mano, vuelva su atención a los usuarios debido a que el objetivo será entender los tipos de funcionalidad por crear en el sistema.

En el mundo real, entrevistaría a los usuarios. En este ejemplo, basará sus ideas en cierto conocimiento general de las LANs y del dominio. No obstante, tenga presente que en el análisis de sistemas del mundo real, nada puede sustituir a las entrevistas con las personas.

Un grupo de usuarios serán consultores, otros podrían ser oficinistas. Entre otros usuarios en potencia se encontrarán funcionarios corporativos, vendedores, administradores de red, administradores de oficina y administradores de proyectos. (¿Se le ocurren otros?)

En este punto, sería conveniente a mostrar a los usuarios en una jerarquía de generalización, como se observa en la figura 7.8.

## Comprensión de los casos de uso

¿Qué hay de los casos de uso? Hay algunas posibilidades: "Establecer niveles de seguridad", "Crear una propuesta", "Almacenar una propuesta", "Utilizar correo electrónico", "Compartir información de la base de datos", "Realizar la contabilidad", "Conectarse a la LAN desde fuera de ella", "Conectarse a Internet", "Compartir información de la base

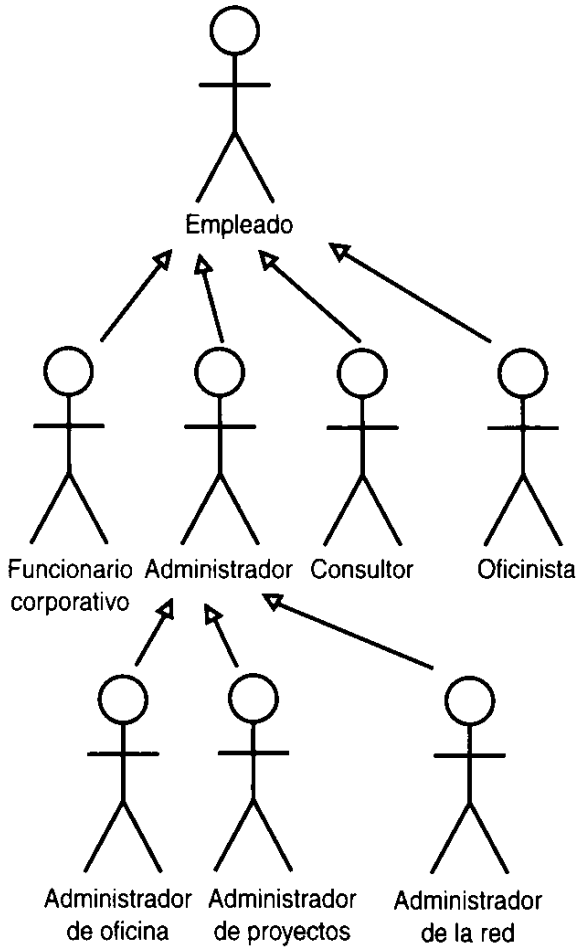
www.elsolucionario.net

www.elsolucionario.net

de datos", "Indizar las propuestas", "Utilizar propuestas previas" y "Compartir impresoras". De acuerdo con esta información, la figura 7.9 le muestra el diagrama de casos de uso de alto nivel que hemos generado.

**FIGURA 7.8**

*La jerarquía de usuarios que interactuarán con la LAN.*




Este conjunto de casos de uso constituye los requerimientos funcionales de la LAN.

## Profundización

Elaboremos uno de los casos de uso de alto nivel y generemos un modelo de caso de uso. Una actividad extremadamente importante en una firma de consultoría es la generación de propuestas, así que examinemos el caso de uso "Crear una propuesta".

Las entrevistas con los consultores probablemente le indicarán cuántos pasos se necesitan en este caso de uso. Para empezar, el actor inicial es un consultor. El consultor tiene que iniciar una sesión en la LAN y ser verificado como usuario válido. Luego tendrá que utilizar algún software integrado para oficina (procesador de textos, hoja de cálculo y gráficos) para escribir la propuesta. En el proceso, el consultor podría volver a utilizar porciones de propuestas previas. La firma de consultoría podría tener una directiva de que un funcionario corporativo y otros dos consultores revisen una propuesta antes de que llegue a manos del cliente. Para ello, el consultor almacena la propuesta en un área central accesible mediante la LAN, y envía a los correos electrónicos de los tres revisores un mensaje que indique que la propuesta se encuentra lista, así como su ubicación.

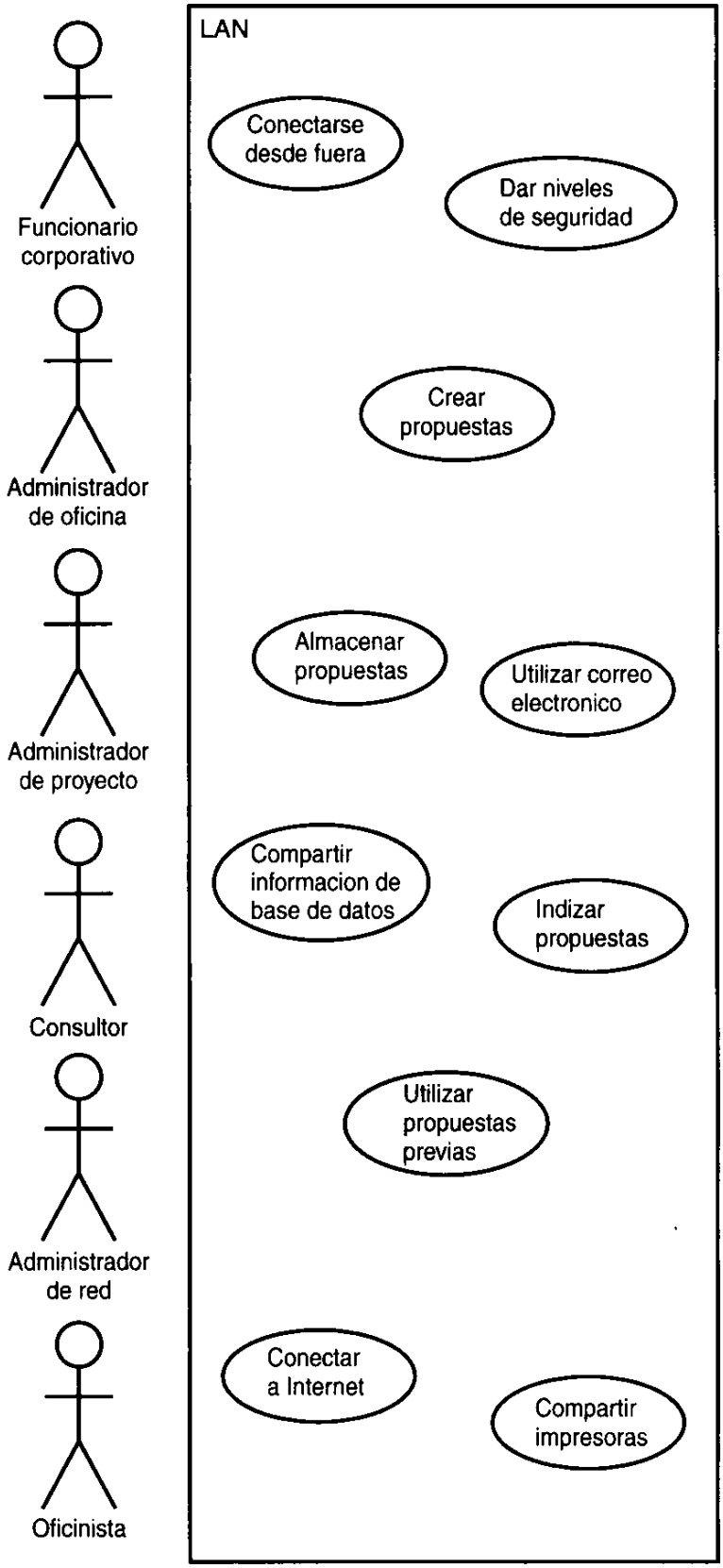
www.elsolucionario.net

www.elsolucionario.net

Luego de recibir los comentarios y hacer las modificaciones necesarias (nuevamente, con el software integrado para oficina), el consultor imprime la propuesta y la envía por correo al cliente. Cuando todo termina, el consultor se retira de la red. El consultor habrá completado una propuesta y es el actor que se beneficia del caso de uso.

**FIGURA 7.9**

*Un diagrama de casos de uso de alto nivel que representa una LAN para una firma de consultoría.*




www.elsolucionario.net

www.elsolucionario.net

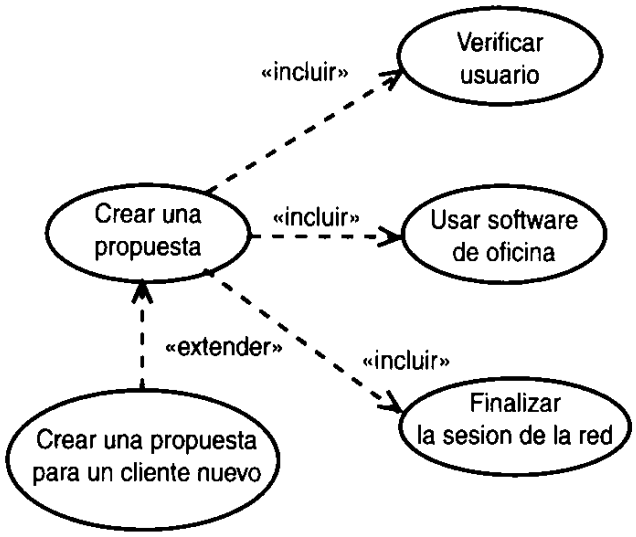
En la secuencia anterior, es claro que ciertos pasos se repetirán de un caso de uso a otro, y ello le llevará a otros casos de uso (posiblemente incluidos) en los que tal vez no había pensado. Iniciar una sesión y ser verificado son dos pasos que pueden incluir varios casos de uso. Por ello, creará un caso de uso "Verificar usuario" que incluya "Crear una propuesta". Otro par de casos de uso son "Utilizar software de oficina" y "Finalizar sesión de la red".

Podrían aparecer otros detalles del proceso de una propuesta cuando vea que las propuestas elaboradas para los clientes nuevos son diferentes a las de los clientes constantes. En sí, las propuestas a los nuevos clientes probablemente incluyen información promocional de la empresa. Con los clientes constantes, no será necesario enviar tal información. Así pues, otro nuevo caso de uso, "Crear una propuesta para un cliente nuevo" extenderá a "Crear una propuesta".

La figura 7.10 le muestra el diagrama de casos de uso que resulta de este análisis del caso de uso "Crear una propuesta".

**FIGURA 7.10**

*El caso de uso "Crear una propuesta" en la LAN de una firma de consultoría.*




Este ejemplo aterriza un punto importante, uno que había destacado anteriormente: El análisis del caso de uso describe el comportamiento de un sistema. No toca a la implementación. ¡Esto es particularmente importante en este caso, dado que el diseño de una LAN supera, por mucho, el alcance de este libro!

# Dónde estamos

Este es un buen momento para ver la estructura general del UML dado que ya ha avanzado en dos de sus principales aspectos: la orientación a objetos y el análisis de casos de uso. Ha visto sus fundamentos y simbología, así como explorado algunas aplicaciones.

www.elsolucionario.net

www.elsolucionario.net

En las horas 2 a la 7 ha trabajado con:

| | |
|---|---|
| Clases | Agregaciones |
| Objetos | Composiciones |
| Interfaces | Estereotipos |
| Casos de uso | Restricciones |
| Actores | Notas |
| Asociaciones | Paquetes |
| Generalizaciones | Extensiones |
| Realizaciones | Inclusiones |

Intentemos dividir este conjunto de elementos en categorías.

## Elementos estructurales

Las clases, objetos, actores, interfaces y casos de uso son cinco de los elementos estructurales en el UML. Aunque tienen diversas diferencias (mismas que, como ejercicio, deberá indicar), son similares en el sentido de que representan partes ya sea físicas o conceptuales de un modelo. Conforme avance en la parte I, verá otros elementos estructurales.

## Relaciones

La asociación, generalización, dependencia y realización, son las relaciones en el UML. (La inclusión y extensión son dos tipos de dependencias.) Sin las relaciones, los modelos UML no serían más que listas de elementos estructurales. Las relaciones conectan a tales elementos y de ese modo conectan los modelos con la realidad.

## Agrupamiento

El paquete es el único elemento de agrupamiento en el UML, éste le permite organizar los elementos estructurales en un modelo. Un paquete puede contener cualquier tipo de elemento estructural, y diferentes tipos a la vez.

## Anotación

La nota es el elemento de anotación del UML; éstas le permiten adjuntar restricciones, comentarios, requerimientos y gráficos explicativos a sus modelos.

www.elsolucionario.net

www.elsolucionario.net

## Extensión

Los estereotipos o clisés son dos estructuras que el UML proporciona para extender el lenguaje. Le permiten crear nuevos elementos además de los existentes, de modo que pueda modelar de forma adecuada la sección de realidad en la que se centrará su sistema.

## ...y más

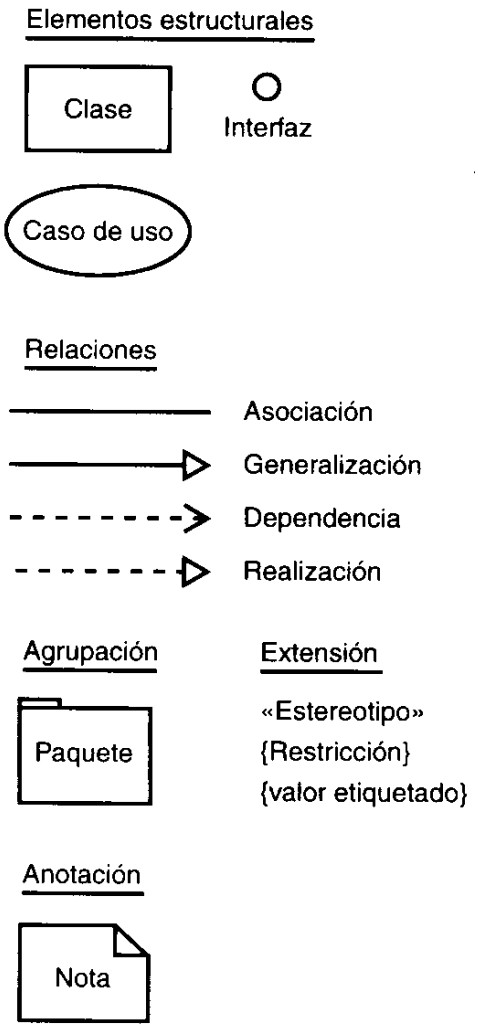
Además de los elementos estructurales, relaciones, agrupamientos, anotaciones y extensiones, el UML cuenta con otra categoría: elementos de comportamiento. Tales elementos le muestran la forma en que las partes de un modelo (como los objetos) cambian con el tiempo. Aún no sabe utilizarlos, pero los verá en la siguiente hora.

# El Panorama

Ahora ya tiene una idea de la forma en que el UML se organiza. La figura 7.11 esquematiza esta organización por usted. Conforme vea las siguientes horas de la parte I, tenga esta organización en mente. Le hará adiciones conforme avance y este "panorama" le mostrará dónde agregar el nuevo conocimiento que adquiera.

**FIGURA 7.11**

*La organización del UML, en términos de los elementos que ha utilizado hasta ahora.*




www.elsolucionario.net

# Resumen

El caso de uso es una poderosa herramienta para obtener los requerimientos funcionales. Los diagramas de casos de uso agregan mayor poder: debido a que conciben los casos de uso, facilitan la comunicación entre los analistas y los usuarios, y entre los analistas y los clientes. En un diagrama, el símbolo del caso de uso es una elipse. El símbolo de un actor es una figura adjunta. Una línea asociativa conecta a un actor con el caso de uso. Los casos de uso están, por lo general, dentro de un rectángulo que representan el confín del sistema.

La inclusión se representa por una línea de dependencia con un estereotipo «incluir». La extensión se representa por una línea de dependencia con un estereotipo «extender». Las otras dos relaciones entre casos de uso son generalización, en la que un caso de uso hereda el sentido y acciones de otro, y el agrupamiento, mismo que organiza un conjunto de casos de uso. La generalización se representa por la misma línea que muestra la herencia entre clases. El agrupamiento se representa por el icono del paquete.

Los diagramas de casos de uso figuran con fuerza en el proceso de análisis. Se empieza con entrevistas con los clientes para obtener diagramas de clases. Éstos proporcionan una base para entrevistar a los usuarios. Tales entrevistas dan por resultado un diagrama de casos de uso de alto nivel que muestra los requerimientos funcionales del sistema. Para crear los modelos de caso de uso, profundice en cada caso de uso de alto nivel. Los diagramas resultantes de caso de uso darán los fundamentos para el diseño y desarrollo.

Ahora que ha visto la orientación a objetos y los casos de uso, está listo para ver el panorama del UML. Los elementos que ha aprendido de las horas 2 a 7 se encuentran en estas categorías: elementos estructurales, relaciones, organización, anotación y extensión. En la siguiente hora verá un elemento de la categoría restante: elementos de comportamiento. Tenga en mente este panorama para que se le facilite el aprendizaje del UML.

www.elsolucionario.net

# Preguntas y respuestas

**P Me di cuenta que en el diagrama de casos de uso de alto nivel no mostró las asociaciones entre los actores y los casos de uso. ¿A qué se debe?**

**R** El diagrama de casos de uso de alto nivel surge en las etapas iniciales de las entrevistas con los usuarios. En este punto, esto es más o menos un ejercicio de recopilación de ideas y el objetivo es encontrar los requerimientos generales, ámbito y confines del sistema. Las asociaciones tendrán mayor sentido cuando posteriores entrevistas con los clientes le lleven a profundizar en cada requerimiento y que los modelos de casos de uso tomen forma.

www.elsolucionario.net

**P ¿Por qué es importante tener en cuenta tal "panorama" del UML? ¿No bastaría con que sepa utilizar cada tipo de diagrama?**

**R** Si usted comprende la organización del UML, podrá manejar situaciones que no haya encontrado antes. Podrá reconocer cuando un elemento UML existente no haga el trabajo, y sabrá cómo construir uno nuevo. También sabrá cómo crear un diagrama híbrido si llegara a ser la única forma de presentar claramente un modelo.

# Taller

En este taller continuará con el conocimiento obtenido en la hora 6 y lo usará como base para el conocimiento de la hora 7. El objetivo es utilizar su nuevo conocimiento para concebir los casos de uso y sus relaciones. Las respuestas aparecen en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. Mencione dos ventajas de concebir un caso de uso.
2. Describa la generalización y el agrupamiento, las relaciones entre los casos de uso que ha visto durante esta hora. Mencione dos situaciones en las que usted agruparía los casos de uso.
3. ¿Cuáles son las similitudes entre las clases y los casos de uso? ¿Cuáles las diferencias?

## Ejercicios

1. Bosqueje el diagrama de un modelo de caso de uso para un control remoto de una televisión. Asegúrese de incluir todas las funciones del control remoto como casos de uso para su modelo.
2. En el segundo ejercicio de la hora 6 indicó a los actores y casos de uso de un almacén de cómputo. Esta vez, dibuje un diagrama de casos de uso de alto nivel con base en el trabajo que realizó en tal ejercicio. Luego, genere un modelo de caso de uso para al menos uno de los casos de uso de alto nivel. En su trabajo, intente incorporar las relaciones «incluir» o «extender» que sean necesarias.

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 8

# Diagramas de estados

Hasta ahora ha comprendido los importantes elementos estructurales del UML. Ahora verá un elemento que le muestra cómo modificar los procedimientos con el tiempo.

En esta hora se tratarán los siguientes temas:

- Qué es un diagrama de estados
- Sucesos, acciones y condiciones de seguridad
- Subestados: secuenciales y concurrentes
- Estados históricos
- Por qué son importantes los diagramas de estados
- Adición del diagrama de estados al panorama del UML

**TÉRMINO NUEVO** Al finalizar la hora anterior, dije que aquí trataría una nueva categoría de elementos con la cual no había trabajado, el elemento de comportamiento, éste muestra la forma en que las partes de un modelo UML cambian con el tiempo. Verá un miembro en particular de esta categoría, el diagrama de estados.

www.elsolucionario.net

www.elsolucionario.net

Cada año trae consigo nuevos estilos en ropas y automóviles, las estaciones cambian el color de las hojas de los árboles y cada año que pasa deja entrever el crecimiento y madurez de los niños. Al pasar el tiempo y conforme suceden las cosas, hay cambios que afectan a los objetos que nos rodean.

Esto también se aplica en cualquier sistema. Conforme el sistema interactúa con los usuarios y (posiblemente) con otros sistemas, los objetos que lo conforman pasarán por cambios necesarios para ajustar las interacciones. Si va a modelar sistemas, necesitará contar con un mecanismo para los cambios en el modelo.

# Qué es un diagrama de estados

Una manera para caracterizar un cambio en un sistema es decir que los objetos que lo componen modificaron su *estado* como respuesta a los sucesos y al tiempo. He aquí algunos ejemplos rápidos:

Cuando acciona el interruptor, la fuente de luz cambia su estado de apagada a encendida.

Cuando presiona un botón de un control remoto, una televisión cambia su estado para mostrarle un canal u otro.

Luego de un lapso adecuado, una lavadora cambia su estado de "lavar" a "enjuagar".

TERMINO NUEVO El *diagrama de estados* UML captura este tipo de cambios. Presenta los estados en los que puede encontrarse un objeto junto con las transiciones entre los estados, y muestra los puntos inicial y final de una secuencia de cambios de estado.

TERMINO NUEVO Un diagrama de estados también se conoce como un *motor de estado*.

Tenga en cuenta que un diagrama de estados es intrínsecamente distinto, de manera muy significativa, de uno de clase, de objeto o de un caso de uso. Los diagramas que ya ha visto modelan el comportamiento de un sistema, o al menos un grupo de clases, objetos o casos de uso. Un diagrama de estados muestra las condiciones de un solo objeto.

## Simbología

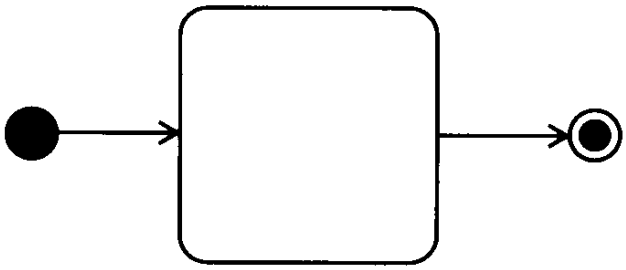
La figura 8.1 le muestra el rectángulo de vértices redondeados que representa a un estado, junto con una línea continua y una punta de flecha, mismas que representan a una transición. La punta de la flecha apunta hacia el estado donde se hará la transición. La figura también muestra un círculo relleno que simboliza un punto inicial y la diana que representa a un punto final.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 8.1**

*Los símbolos UML en un diagrama de estados. El icono para el estado es un rectángulo de vértices redondeados, y el símbolo de una transición es una línea continua y una punta de flecha. Un círculo relleno se interpreta como el punto inicial de una secuencia de estados, y una diana representa al punto final.*



## Adición de detalles al icono de estado

El UML le da la opción de agregar detalles a la simbología. Así como es posible dividir un símbolo de clase en tres áreas (nombre, atributos y operaciones), puede dividir el icono de estado de igual forma. El área superior contendrá el nombre del estado (que tiene que establecer ya sea que haya la subdivisión o no), el área central contendrá las variables de estado, y el área inferior las actividades. La figura 8.2 le muestra estos detalles.

**FIGURA 8.2**

*Puede subdividir el símbolo del estado en áreas que muestren el nombre, variables y actividades del estado.*

Las variables de estado como cronómetros o contadores son, en ocasiones, de ayuda. Las actividades constan de sucesos y acciones: tres de las más utilizadas son *entrada* (qué sucede cuando el sistema entra al estado), *salida* (qué sucede cuando el sistema sale del estado), y *hacer* (qué sucede cuando el sistema está en el estado). Puede agregar otros conforme sea necesario.

Un máquina de fax sirve como ejemplo de un objeto que puede pasar por diversas variables y actividades de estado. Cuando se envía un fax —esto es, cuando se encuentra en estado de envío de fax— la máquina de fax anota la fecha y hora en que inició el envío (los valores de las variables de estado "fecha" y "hora"), y también anota su número telefónico así como el nombre del propietario (los valores de las variables de estado "teléfono" y "propietario"). Al encontrarse en este estado, la máquina se encarga de agregar un registro de fecha y hora al fax, número telefónico y nombre del propietario. En otras actividades de este estado, la máquina jalará las hojas, paginará el fax y finalizará la transmisión.

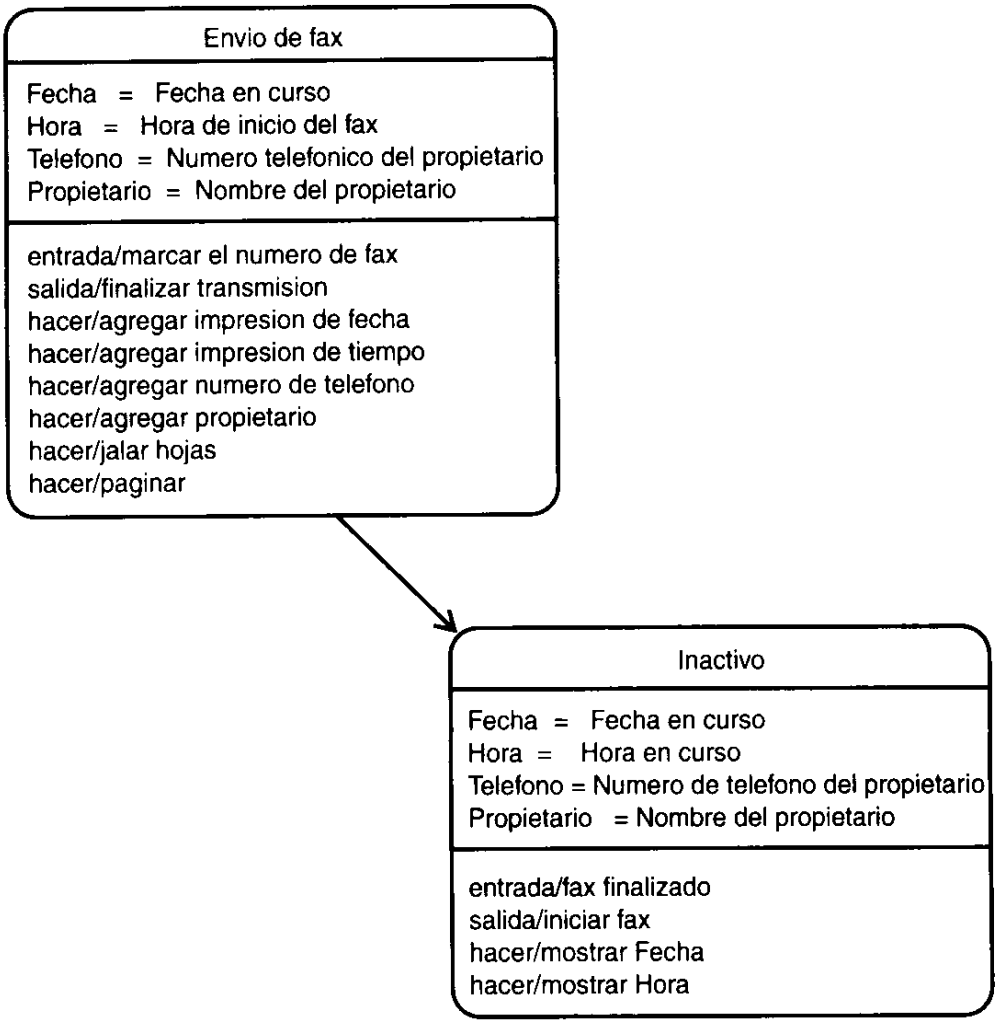
www.elsolucionario.net

www.elsolucionario.net

Mientras se encuentre en el estado de inactividad, la máquina de fax mostrará la fecha y la hora en una pantalla. La figura 8.3 le muestra el diagrama de estados.

**FIGURA 8.3**

*La máquina de fax es un buen ejemplo de un estado con variables y actividades.*




## Sucesos y acciones

**TÉRMINO NUEVO**

También puede agregar ciertos detalles a las líneas de transición. Puede indicar un suceso que provoque una transición (*desencadenar un suceso*), y la actividad de cómputo (*la acción*) que se ejecute y haga que suceda la modificación del estado. A los sucesos y acciones los escribirá cerca de la línea de transición mediante una diagonal para separar un suceso desencadenado de una acción. En ocasiones un evento causará una transición sin una acción asociada, y algunas veces una transición sucederá dado que un estado finalizará una actividad (en lugar de hacerlo por un suceso). A este tipo de transición se le conoce como *transición no desencadenada*. La GUI (interfaz gráfica de usuario) con que interactúe le dará ejemplos de detalles de la transición. Por el momento, asumamos que la GUI puede establecerse en uno de tres estados:

Inicialización

Operación

Apagar

www.elsolucionario.net

www.elsolucionario.net

Cuando encienda su equipo, se ejecutará un proceso de arranque. Al encender la PC se desencadena un suceso que provoca que la GUI aparezca luego de una transición desde el estado de Inicialización, y el arranque es una acción que se realiza durante tal transición.

Como resultado de las actividades en el estado de inicialización, la GUI entra al modo de Operación. Cuando desea apagar su PC, desencadena un suceso que provoca la transición hacia el estado de Apagado, y con ello la PC se apaga. La figura 8.4 muestra el diagrama de estados que captura los estados y transiciones en la GUI.

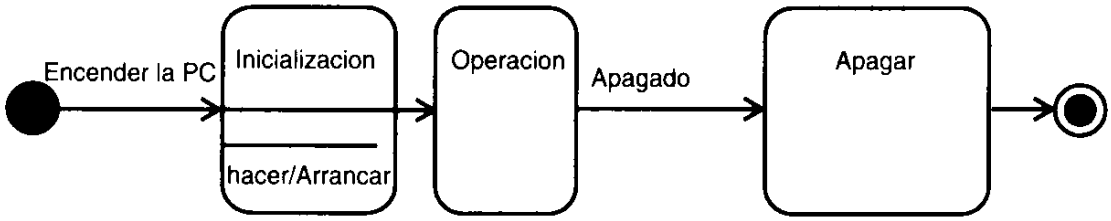



**FIGURA 8.4**

*Los estados y transiciones de una interfaz gráfica del usuario incluyen el desencadenamiento de eventos, acciones y transiciones no desencadenadas.*

## Condiciones de seguridad

La anterior secuencia de estados de la GUI deja mucho que desear. Por ejemplo: si deja solo su equipo o si realizara alguna actividad en la que no tocará al ratón o al teclado, podría aparecer un protector de pantallas que evitaría el desgaste de su pantalla. Para decir esto en términos de cambio de estado, si ha pasado cierto tiempo sin que haya interacción con el usuario, la GUI hará una transición del estado Operación a un estado que no aparece en la figura 8.4: el estado de Protector de pantallas.

El intervalo se especifica en el panel de control de su sistema operativo (Windows en este caso). Por lo general es de 15 minutos. Cualquier opresión de una tecla o movimiento del ratón provocará una transición del estado Protector de pantallas al estado Operación.

**TERMINO NUEVO**

El intervalo de 15 minutos es una *condición de seguridad*: cuando se llega a ella, se realiza la transición. La figura 8.5 muestra el diagrama de estados de la GUI con el estado Protector de pantalla y la condición de seguridad añadida. Vea que la condición de seguridad se establece como expresión booleana.

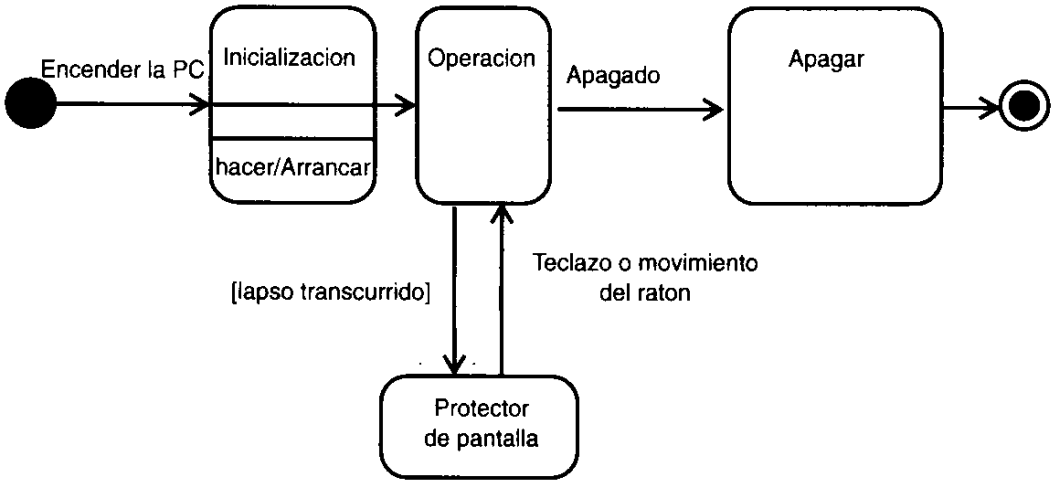



**FIGURA 8.5**

*El diagrama de estados para la GUI, con el estado Protector de pantalla y la condición de seguridad.*

www.elsolucionario.net

www.elsolucionario.net

# Subestados

Nuestro modelo de la GUI aún está algo vacío. El estado Operación, en particular, es mucho más sustancioso de lo que he indicado en las figuras 8.4 y 8.5.

Cuando la GUI está en el estado Operación, hay muchas cosas que ocurren tras bambalinas, aunque no sean particularmente evidentes en la pantalla. La GUI aguarda de forma constante a que usted haga algo (oprimir una tecla, mover el ratón u oprimir uno de sus botones). Luego, deberá registrar tales acciones y modificar lo que se despliega para reflejarlas en la pantalla (como mover el cursor cuando usted mueva el ratón, o mostrar una "a" cuando usted oprima la tecla "a").

**TÉRMINO NUEVO**

Con ello la GUI atravesará por varios cambios mientras se encuentre en el estado Operación. Tales cambios serán cambios de estado. Dado que estos estados se encuentran dentro de otros, se conocerán como *subestados*. Hay dos tipos de subestados: *secuencial* y *concurrente*.

## Subestados secuenciales

Como su nombre lo indica, los subestados secuenciales suceden uno detrás de otro. Si retomamos los subestados mencionados con anterioridad dentro del estado Operación de la GUI, tendrá la siguiente secuencia:

A la espera de acción del usuario

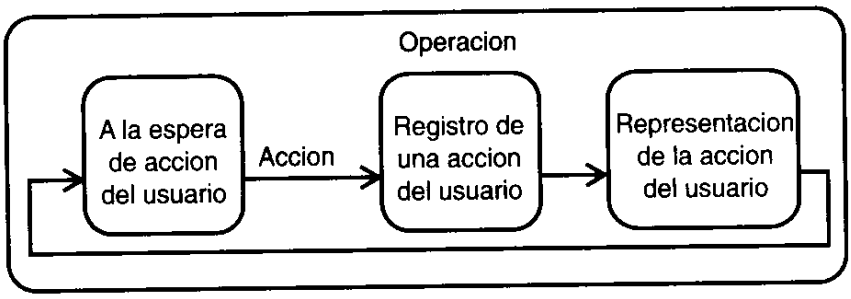
Registro de una acción del usuario

Representación de la acción del usuario

La acción del usuario desencadena la transición a partir de A la espera de acción del usuario hacia Registro de una acción del usuario. Las actividades dentro del Registro trascienden de la GUI hacia la Representación de la acción del usuario. Después del tercer estado, la GUI vuelve a iniciar A la espera de acción del usuario. La figura 8.6 le muestra cómo representar los subestados secuenciales dentro del estado Operación.

**FIGURA 8.6**

*Subestados secuenciales dentro del estado Operación de la GUI.*




## Subestados concurrentes

Dentro del estado Operación, la GUI no sólo aguarda a que usted haga algo. También verifica el cronómetro del sistema y (posiblemente) actualiza el despliegue de una aplicación luego de un intervalo específico. Por ejemplo, una aplicación podría incluir un reloj en pantalla que tuviera que actualizar la GUI.

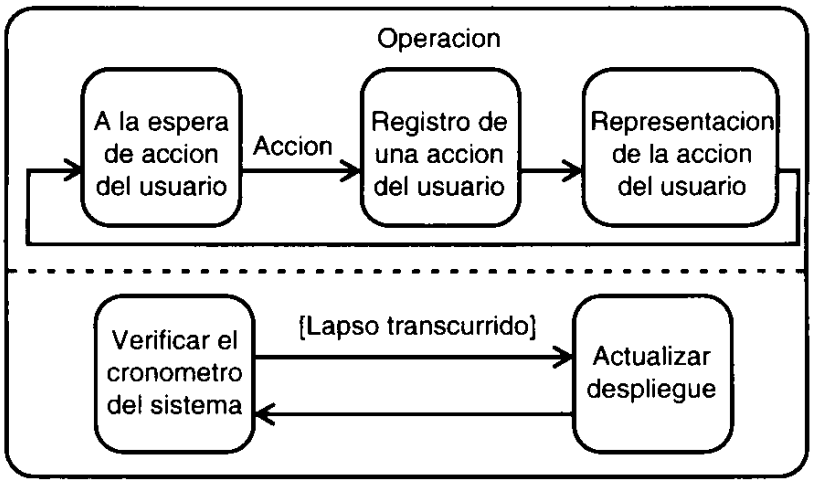
www.elsolucionario.net

www.elsolucionario.net

Todo esto sucede al mismo tiempo que la secuencia que ya indiqué. Aunque cada secuencia es, claro, un conjunto de subestados secuenciales, las dos secuencias son concurrentes entre sí. Puede representar la concurrencia con una línea discontinua entre los estados concurrentes, como en la figura 8.7.

**FIGURA 8.7**

*Los subestados concurrentes suceden al mismo tiempo. Una línea discontinua los separa.*




TERMINO NUEVO

La separación del estado Operación en dos componentes podría recordarle algo. ¿Recuerda cuando traté el tema de las adiciones y composiciones? Cuando cada componente sea parte de un "todo", tratará con una composición. Las partes concurrentes del estado Operación tienen el mismo tipo de relación con él. Por ello, el estado Operación es un *estado compuesto*. Un estado que consta sólo de subestados secuenciales, también es un estado compuesto.

# Estados históricos

Cuando se activa su protector de pantallas y mueve su ratón para regresar al estado Operación ¿qué ocurre? ¿Acaso su pantalla retoma el estado inicial, como si apenas se hubiera encendido? ¿O lucirá tal como la dejó antes de que se activara el protector de pantallas?

Es obvio, si el protector de pantallas provocara que la pantalla regresara a su estado inicial de operación, la idea del protector de pantallas sería contraproducente. Los usuarios podrían perder su trabajo y tendrían que reiniciar una sesión nuevamente.

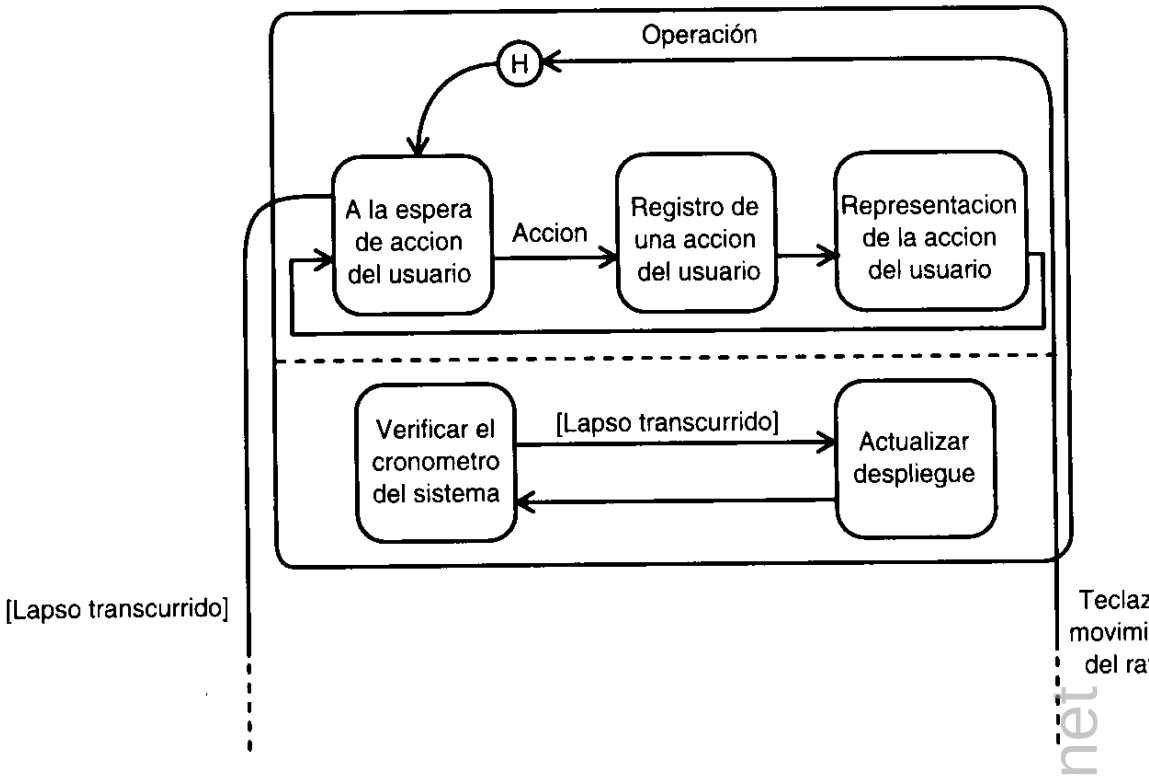
El diagrama de estados históricos captura esta idea. El UML proporciona un símbolo que muestra que un estado compuesto recuerda su subestado activo cuando el objeto trasciende fuera del estado compuesto. El símbolo es la letra "H" encerrada en un círculo que se conecta por una línea continua al subestado por recordar, con una punta de flecha que apunta a tal subestado. La figura 8.8 muestra este símbolo en el estado Operación.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 8.8**

*El estado histórico, simbolizado con la "H" dentro del círculo, le muestra que un estado compuesto recuerda su subestado activo cuando el objeto trasciende fuera de tal estado compuesto.*




**TERMINO NUEVO** En el diagrama de estados no he tratado con las ventanas que están abiertas por otras ventanas (es decir, con subestados anidados en otros). Cuando un estado histórico recuerda los subestados en todos los niveles de anidación (como el estado Operación de Windows lo hace), el estado histórico es *profundo*. Si sólo recuerda el subestado principal, el estado histórico será *superficial*. Un estado histórico profundo se representa agregando un asterisco (*) a la "H" en el círculo (H*).

**TERMINO NUEVO** El estado histórico y el estado inicial (representados por el círculo relleno) son conocidos como *pseudoestados*. No tienen variables de estados ni actividades, por lo que no son estados "completos".

# Mensajes y señales

En el ejemplo, el suceso desencadenado que provocó la transición de Protector de pantalla a Operación es la opresión de una tecla, un movimiento del ratón o una opresión de uno de sus botones. Cualquiera de estos sucesos es, en efecto, un mensaje del usuario a la GUI. Esto es un concepto importante dado que los objetos se comunican mediante el envío de mensajes entre sí. En este caso, el suceso desencadenado es un mensaje de un objeto (el usuario) a otro (la GUI).

www.elsolucionario.net

www.elsolucionario.net

**TÉRMINO NUEVO** Un mensaje que desencadena una transición en el diagrama de estados del objeto receptor se conoce como *señal*. En el mundo de la orientación a objetos, el envío de una señal es lo mismo que crear un objeto Señal y transmitirlo al objeto receptor. El objeto Señal cuenta con propiedades que se representan como atributos. Dado que una señal es un objeto, es posible crear jerarquías de herencia de señales.

# Por qué son importantes los diagramas de estados

El diagrama de estados del UML proporciona una gran variedad de símbolos y abarca varias ideas (todas para modelar los cambios por los que pasa un objeto). Este tipo de diagrama tiene el potencial de convertirse en algo complejo con pasmosa rapidez. ¿En verdad es necesario?

De hecho, así es. Es necesario contar con diagramas de estados dado que permiten a los analistas, diseñadores y desarrolladores comprender el comportamiento de los objetos de un sistema. Un diagrama de clases y el diagrama de objetos correspondiente sólo muestra los aspectos estáticos de un sistema. Muestran las jerarquías y asociaciones, y le indican qué son las operaciones. Pero no le muestran los detalles dinámicos de las operaciones.

Los desarrolladores, en particular, deben saber la forma en que los objetos se supone que se comportarán, ya que son ellos quienes tendrán que establecer tales comportamientos en el software. No es suficiente con implementar un objeto: los desarrolladores deben hacer que tal objeto haga algo. Los diagramas de estados se aseguran que no tendrán que adivinar lo que se supone que harán los objetos. Con una clara representación del comportamiento del objeto, aumenta la probabilidad de que el equipo de desarrollo produzca un sistema que cumpla con los requerimientos.

www.elsolucionario.net

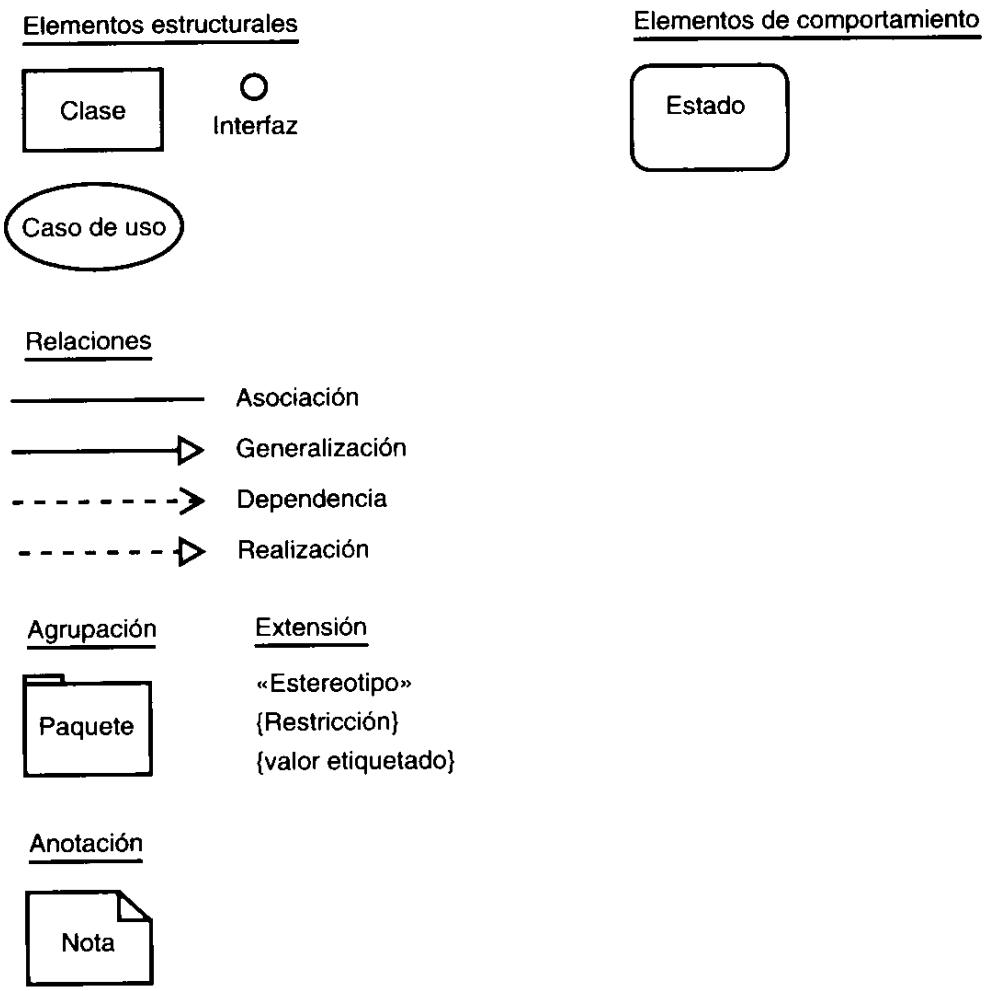
# Adiciones al panorama

Ahora puede agregar los "Elementos de comportamiento" al panorama del UML. La figura 8.9 le muestra la imagen con el diagrama de estados incluido.

www.elsolucionario.net

**FIGURA 8.9**

*El panorama del UML ahora incluye un elemento de comportamiento: el diagrama de estados. La organización del UML, en términos de los elementos que ha utilizado hasta ahora.*




# Resumen

Los objetos en los sistemas modifican sus estados como respuestas a sucesos y al tiempo. El diagrama de estados de UML captura estos cambios de estado. Un diagrama de estados se enfoca en los cambios de estado en un solo objeto. Un rectángulo de vértices redondeados representa a un estado, y una línea continua con una punta de flecha representa una transición de un estado a otro.

El símbolo del estado contiene el nombre del mismo y puede tener variables y actividades del estado. Una transición puede suceder como respuesta a un suceso desencadenado, e implicar una respuesta o acción. Una transición también puede ocurrir por la actividad en un estado: una transición que ocurre de esta forma se conoce como *transición no desencadenada*. Finalmente, una transición puede ocurrir cuando se cumple una condición particular, o *condición de seguridad*.

En ocasiones, un estado consta de subestados. Los subestados pueden ser secuenciales (ocurrir uno después del otro) o concurrentes (ocurrir al mismo tiempo). Un estado que consta de subestados se conoce como estado compuesto. Un estado histórico indica que un estado compuesto recordará su subestado cuando el objeto trascienda de este estado compuesto. Un estado histórico puede ser superficial o profundo. Tales términos son propios de los subestados anidados. Un estado histórico *superficial* recuerda sólo el subestado principal. Un estado histórico *profundo* recuerda todos los niveles de los subestados.

www.elsolucionario.net

www.elsolucionario.net

Cuando un objeto envía un mensaje que desencadena una transición en el diagrama de estados de otro objeto, tal mensaje es una *señal*. Una señal es, por sí misma, un objeto, y podrá crear una jerarquía de herencia de las señales.

Es necesario contar con los diagramas de estados porque facilitan la comprensión de los objetos de un sistema a los analistas, diseñadores y desarrolladores. Los desarrolladores, en particular, deben saber cómo se supone que se comportarán los objetos, dado que serán quienes tengan que establecer estos comportamientos en el software. No es suficiente implementar un objeto: los desarrolladores tienen que hacer que tal objeto haga algo.

# Preguntas y respuestas

**P ¿Cuál es la mejor manera de empezar a crear un diagrama de estados?**

**R** Es muy parecido a crear un diagrama de clases o un modelo de caso de uso. En el diagrama de clases, listará todas las clases y luego realizará las asociaciones entre ellas. En el diagrama de estados, primero listará los estados del objeto, y luego se enfocará en las transiciones. Conforme avance en cada transición, deberá prever si un suceso desencadenado lo activará y si se realizará alguna acción.

**P ¿Cada diagrama de estados debe tener un estado final (el que se representa por la diana)?**

**R** No. Un objeto que nunca queda inactivo jamás tendrá un estado final.

**P ¿Alguna sugerencia para diseñar un diagrama de estados?**

**R** Intente arreglar los estados y transiciones para minimizar el cruzamiento de líneas. Uno de los objetivos de este diagrama (y de cualquier otro) se centra en la claridad. Si las personas no pueden comprender los modelos que cree, nadie los usará y sus esfuerzos (no importa qué tan minuciosos hayan sido) habrán sido infructuosos.

www.elsolucionario.net

# Taller

El cuestionario y los ejercicios le harán trascender al estado "Aprendizaje de diagramas de estados". Como siempre, encontrará las respuestas en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionarios

1. ¿De qué forma difiere un diagrama de estados de uno de clases, de objetos o de caso de uso?
2. Defina los siguientes términos: *transición*, *suceso* y *acción*.
3. ¿Qué es una *transición no desencadenada*?
4. ¿Cuál es la diferencia entre los subestados secuenciales y los concurrentes?

www.elsolucionario.net

## Ejercicios

1. Suponga que diseñará un tostador. Cree el diagrama de estados que controle los estados del pan en el tostador. Incluya los sucesos desencadenados, acciones y condiciones de seguridad necesarios.
2. Cada vez que un objeto envíe una señal, se creará un objeto Señal y será transmitido. En Windows, hay varias señales posibles a partir de la GUI. Suponga que la señal (el tipo de señal que envíe a Windows) sea una clase. (¿Qué tipo de clase es?) Cree un diagrama de clases de las posibles señales y muestre toda la herencia inherente.
3. La figura 8.7 le muestra los subestados concurrentes dentro del estado Operación de la GUI. Dibuje un diagrama del estado Protector de pantalla que incluya los subestados concurrentes.

www.elsolucionario.net

www.elsolucionario.net

# HORA 9

# Diagramas de secuencias

Los diagramas de estados se enfocan a los diferentes estados de un objeto. Esto es sólo una pequeña parte del cuadro. El diagrama de secuencias del UML establece el siguiente paso y le muestra la forma en que los objetos se comunican entre sí al transcurrir el tiempo.

En esta hora se tratarán los siguientes temas:

- Qué es un diagrama de secuencias
- La GUI
- Diagramas de instancias y diagramas genéricos
- Uso de "si" y "mientras"
- Creación de un objeto en la secuencia
- Representación de la recursividad
- Diagramas de secuencias en el panorama del UML

Los diagramas de estados que vio en la hora anterior se centran en un objeto y muestran los cambios por los que pasa dicho objeto.

www.elsolucionario.net

www.elsolucionario.net

El UML le permite expandir su campo de visión y le muestra la forma en que un objeto interacciona con otros. En este campo de visión expandido, incluirá una importante dimensión: el tiempo. La idea primordial es que las interacciones entre los objetos se realizan en una secuencia establecida y que la secuencia se toma su tiempo en ir del principio al fin. Al momento de crear un sistema tendrá que especificar la secuencia, y para ello, utilizará al diagrama correspondiente.

# Qué es un diagrama de secuencias

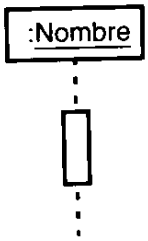
**TERMINO NUEVO** El *diagrama de secuencias* consta de objetos que se representan del modo usual: rectángulos con nombre (subrayado), mensajes representados por líneas continuas con una punta de flecha y el tiempo representado como una progresión vertical.

## Objetos

**TERMINO NUEVO** Los objetos se colocan cerca de la parte superior del diagrama de izquierda a derecha y se acomodan de manera que simplifiquen al diagrama. La extensión que está debajo (y en forma descendente) de cada objeto será una línea discontinua conocida como la *línea de vida* de un objeto. Junto con la línea de vida de un objeto se encuentra un pequeño rectángulo conocido como *activación*, el cual representa la ejecución de una operación que realiza el objeto. La longitud del rectángulo se interpreta como la duración de la activación. La figura 9.1 muestra un objeto, su línea de vida y su activación.

**FIGURA 9.1**

*Representación de un objeto en un diagrama de secuencias.*




## Mensaje

Un mensaje que va de un objeto a otro pasa de la línea de vida de un objeto a la de otro. Un objeto puede enviarse un mensaje a sí mismo (es decir, desde su línea de vida hacia su propia línea de vida).

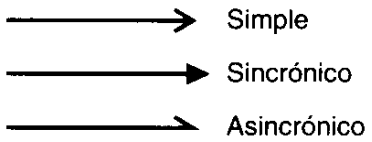
Un mensaje puede ser *simple*, *sincrónico*, o *asincrónico*. Un mensaje simple es la transferencia del control de un objeto a otro. Si un objeto envía un mensaje sincrónico, esperará la respuesta a tal mensaje antes de continuar con su trabajo. Si un objeto envía un mensaje asincrónico, no esperará una respuesta antes de continuar. En el diagrama de secuencias, los símbolos del mensaje varían, por ejemplo, la punta de la flecha de un mensaje simple está formada por dos líneas, la punta de la flecha de un mensaje sincrónico está rellena y la de un asincrónico tiene una sola línea, como se aprecia en la figura 9.2.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 9.2**

*Símbolos para los mensajes en un diagrama de secuencias.*



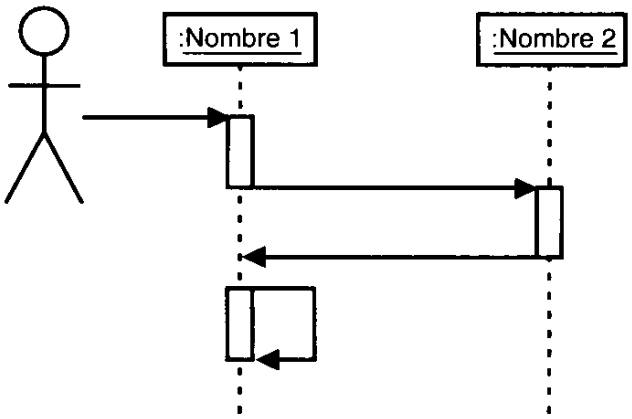

## Tiempo

El diagrama representa al tiempo en dirección vertical. El tiempo se inicia en la parte superior y avanza hacia la parte inferior. Un mensaje que esté más cerca de la parte superior ocurrirá antes que uno que esté cerca de la parte inferior.

Con ello, el diagrama de secuencias tiene dos dimensiones. La dimensión horizontal es la disposición de los objetos, y la dimensión vertical muestra el paso del tiempo. La figura 9.3 muestra al conjunto básico de símbolos del diagrama de secuencias, con los símbolos en funcionamiento conjunto. La figura muestra a un actor que inicia la secuencia, aunque, en sentido estricto, la figura adjunta no es parte del conjunto de símbolos del diagrama de secuencias.

**FIGURA 9.3**

*En un diagrama de secuencias los objetos se colocan de izquierda a derecha en la parte superior. Cada línea de vida de un objeto es una línea discontinua que se desplaza hacia abajo del objeto. Una línea continua con una punta de flecha conecta a una línea de vida con otra, y representa un mensaje de un objeto a otro. El tiempo se inicia en la parte superior y continúa hacia abajo. Aunque un actor es el que normalmente inicia la secuencia, su símbolo no es parte del conjunto de símbolos del diagrama de secuencias.*




Para ver en acción a esta importante herramienta del UML, apliquémosla en los ejemplos que hemos visto en las horas anteriores. Cada aplicación le mostrará algunos conceptos importantes que se relacionan con los diagramas de secuencias.

www.elsolucionario.net

www.elsolucionario.net

# La GUI

En la hora anterior desarrolló los diagramas de estados que muestran los cambios por los que pasa una GUI. Ahora dibujará un diagrama de secuencias que represente las interactividades de la GUI con otros objetos.

## La secuencia

Suponga que el usuario de una GUI presiona una tecla alfanumérica; si asumimos que utiliza una aplicación como un procesador de textos, el carácter correspondiente deberá aparecer de inmediato en la pantalla. ¿Qué ocurre tras bambalinas para que esto suceda?

1. La GUI notifica al sistema operativo que se oprimió una tecla.
2. El sistema operativo le notifica a la CPU.
3. El sistema operativo actualiza la GUI.
4. La CPU notifica a la tarjeta de vídeo.
5. La tarjeta de vídeo envía un mensaje al monitor.
6. El monitor presenta el carácter alfanumérico en la pantalla, con lo que se hará evidente al usuario.

Todo esto ocurre con tanta rapidez que olvidamos que todo ello se realiza. (¡Si acaso pensábamos que ocurría!)

## El diagrama de secuencias

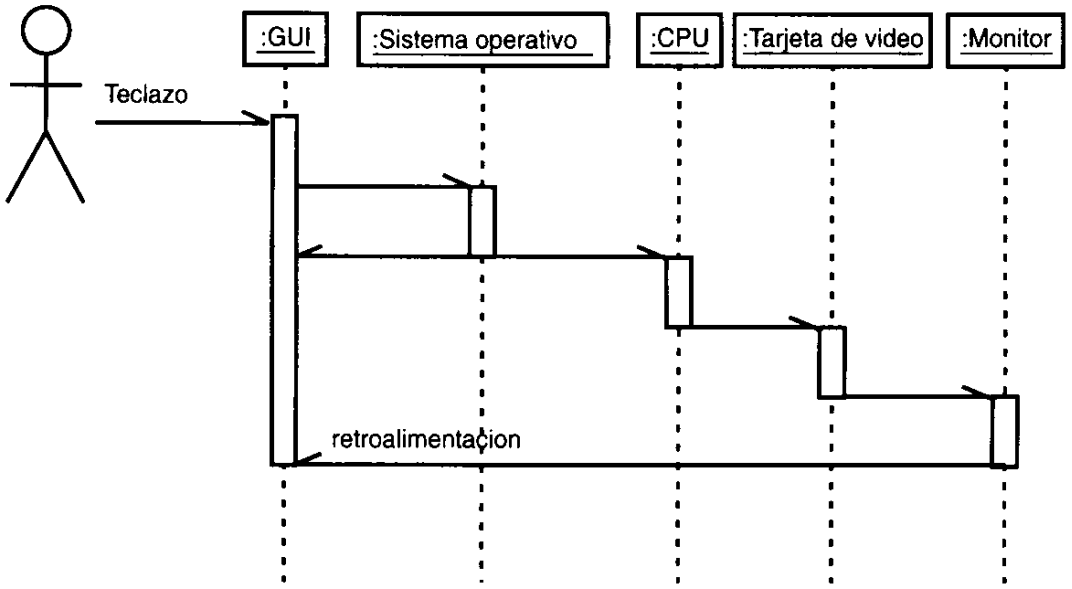
La figura 9.4 representa el diagrama de secuencias de la GUI. Como ve, los mensajes son asincrónicos: ninguno de los componentes aguarda nada antes de continuar. Al trabajar con algunas aplicaciones de Windows, tal vez haya sentido algunos de los efectos de la comunicación asincrónica, particularmente en una máquina lenta. Cuando teclea en un procesador de textos, en ocasiones no ve aparecer en la pantalla el carácter correspondiente a la tecla que haya oprimido sino hasta después de haber oprimido algunas más.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 9.4**

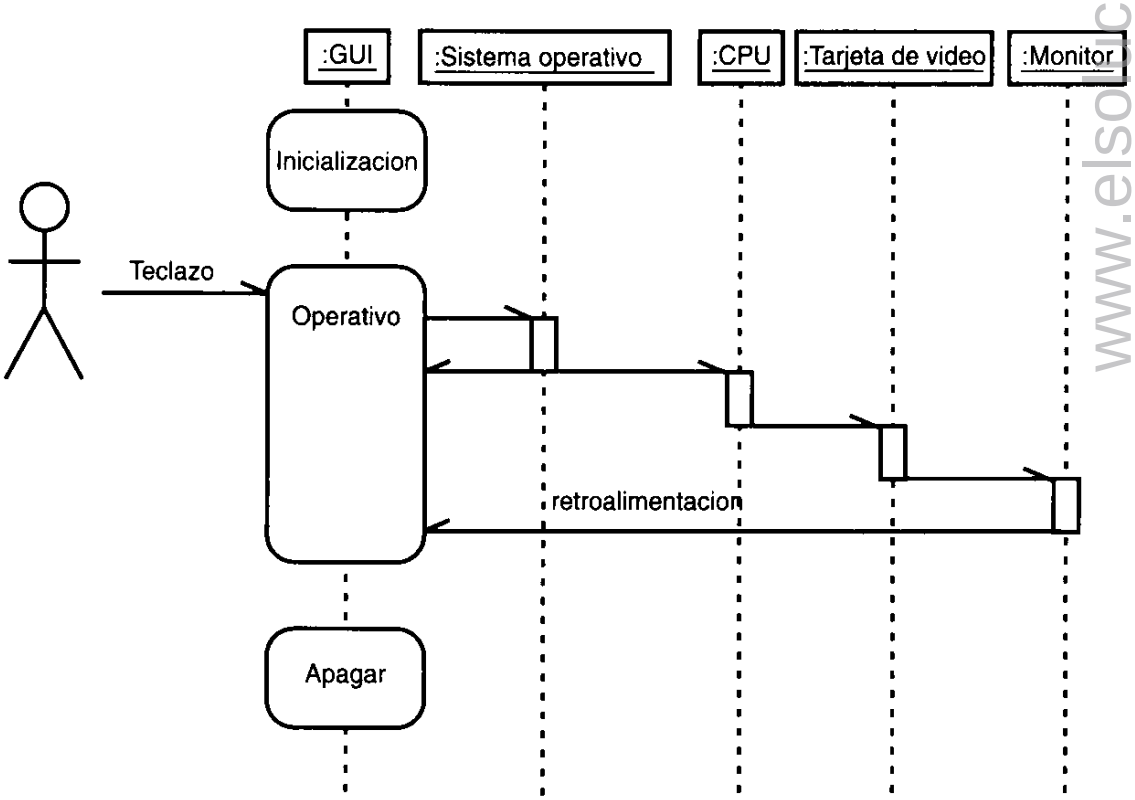
*Un diagrama de secuencias que muestra la forma en que la GUI interacciona con otros objetos.*




En ocasiones, es muy instructivo mostrar los estados de uno o varios de los objetos en el diagrama de secuencias. Dado que ya ha analizado los estados de la GUI (en la hora anterior), esto es fácil de hacer. La figura 9.5 le muestra un híbrido: el diagrama de secuencias de la GUI con los estados de la GUI. Observe que la secuencia se origina y finaliza en el estado Operativo de la GUI, como podría esperarlo.

**FIGURA 9.5**

*Un diagrama de secuencias puede mostrar los estados de un objeto.*




www.elsolucionario.net

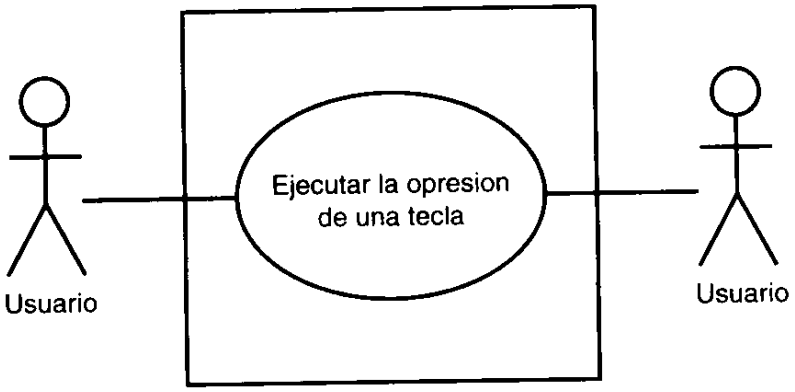
En un diagrama de secuencias, otra forma de mostrar el cambio de estado de un objeto es incluir al objeto más de una vez en el diagrama.

## El caso de uso

¿Qué es exactamente lo que representa un diagrama de secuencias? En este ejemplo, muestra las interacciones de objetos que se realizan durante un escenario sencillo: la opresión de una tecla. Este escenario podría ser parte de un caso de uso llamado "Ejecutar la opresión de una tecla" (vea la figura 9.6). Al representar gráficamente las interacciones del sistema en el caso de uso, el diagrama de secuencias habrá, en efecto, "delineado" el caso de uso dentro del sistema.

**FIGURA 9.6**

*El caso de uso representado gráficamente por el diagrama de secuencias de la figura 9.4.*




En el siguiente ejemplo, examinaré detalladamente la relación entre los casos de uso y los diagramas de secuencias.

# Instancias y genéricos

El ejemplo anterior comenzó con un diagrama de estados. Este otro ejemplo empieza con un caso de uso. "Comprar gaseosa" fue uno de los casos de uso del ejemplo de la máquina de gaseosas en las horas 6, "Introducción a los casos de uso", y 7, "Diagramas de casos de uso".

## Un diagrama de secuencias de instancias

En el mejor escenario del caso de uso "Comprar gaseosa", recuerde que el actor es un cliente que desea adquirir una lata de gaseosa. El cliente inicia el escenario mediante la inserción de dinero en la máquina. Luego hace una selección. Dado que hablamos del mejor escenario, la máquina tiene al menos una lata de la gaseosa elegida y por lo tanto presenta una lata fría al cliente.

Asumamos que en la máquina de gaseosas hay tres objetos que realizan la tarea que nos ocupa: la fachada (la interfaz que la máquina de gaseosas presenta al usuario), el registrador de dinero (que lo recolecta), y el dispensador (que entrega la gaseosa). También daremos por hecho que el registrador de dinero controlará al dispensador. La secuencia será como sigue:

www.elsolucionario.net

www.elsolucionario.net

1. El cliente inserta el dinero en la alcancía que se encuentra en la fachada de la máquina.
2. El cliente hace su elección.
3. El dinero viaja hacia el registrador.
4. El registrador verifica si la gaseosa elegida está en el dispensador.
5. Dado que es el mejor escenario, asumimos que sí hay gaseosas, y el registrador actualiza su reserva de efectivo.
6. El registrador hace que el dispensador entregue la gaseosa en la fachada de la máquina.

**TÉRMINO NUEVO** Dado que el diagrama de secuencias sólo se centra en un escenario (una instancia) en el caso de uso "Comprar gaseosa", se conoce como *diagrama de secuencias de instancias*. La figura 9.7 le muestra este diagrama. Vea que el diagrama muestra mensajes sencillos. Cada mensaje mueve el flujo de control de un objeto a otro.

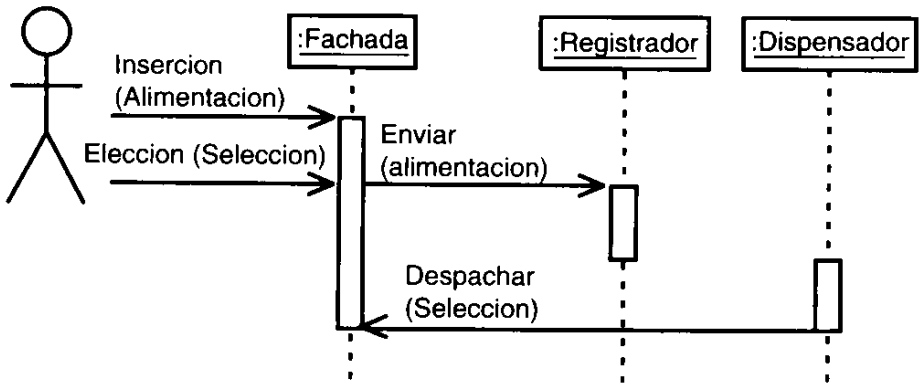



**FIGURA 9.7**
*Este diagrama de secuencias modela tan sólo el mejor escenario del caso de uso "Comprar gaseosa". Por lo tanto, es un diagrama de secuencias de instancias.*

## Un diagrama de secuencias genérico

Como recordará, el caso de uso "Comprar gaseosa" tenía dos escenarios alternos. Uno de ellos se refería al hecho de que la máquina no tuviera la gaseosa seleccionada y el otro cuando el cliente no contaba con el dinero exacto. Si tomara en cuenta todos los escenarios de un caso de uso al momento de crear un diagrama de secuencias, se trataría de un *diagrama de secuencias genérico*.

En este caso podrá generar el diagrama de secuencias genérico a partir del diagrama de secuencias de instancias. Para ello tendrá que justificar el control del flujo. Esto es, tendrá que representar las condiciones y consecuencias de "Monto incorrecto" y "Sin gaseosa".

Para el escenario relacionado con "Monto incorrecto".

1. El registrador verifica si la alimentación del usuario concuerda con el precio de la gaseosa.
2. Si el monto es mayor que el precio, el registrador calcula la diferencia y verifica si cuenta con cambio.

www.elsolucionario.net

www.elsolucionario.net

3. Si se puede devolver la diferencia, el registrador devuelve el cambio al cliente y todo transcurre como antes.
4. Si la diferencia no se encuentra en la reserva del cambio, el registrador regresará el monto alimentado y mostrará un mensaje que indique al cliente que inserte el monto exacto.
5. Si la cantidad insertada es menor al precio, el registrador no hace nada y la máquina esperará más dinero.

Si diseña una máquina de gaseosas para un cliente, tal vez tenga que tomar una decisión de diseño respecto al paso 5. Podrá hacer que la máquina aguarde cierto tiempo, calcule la diferencia entre el precio y el monto insertado, y que muestre un mensaje que solicite al cliente que inserte la diferencia.

Como parte de la decisión, tendrá que responder a estas preguntas: ¿Qué tanto le importará esta facultad al cliente? ¿Cuánto costaría implementar la tecnología que lograra lo anterior?

Éste es un buen ejemplo de la forma en que un diagrama de secuencias puede influir en el proceso de análisis.

Para representar cada condición en la secuencia, tal condición la colocará en un "si" (un si condicional) entre corchetes. Arriba de las flechas de mensaje apropiadas, agregue [alimentación > precio], [alimentación – precio no presente] y [alimentacion – precio presente].

Cada condición causará una bifurcación del control en el mensaje, que separará al mensaje en rutas distintas. Como cada ruta irá al mismo objeto, la bifurcación causará una "ramificación" del control en la línea de vida del objeto receptor, y separará las líneas de vida en rutas distintas. En algún lugar de la secuencia, las ramas del mensaje confluirán, como las bifurcaciones en las líneas de vida.

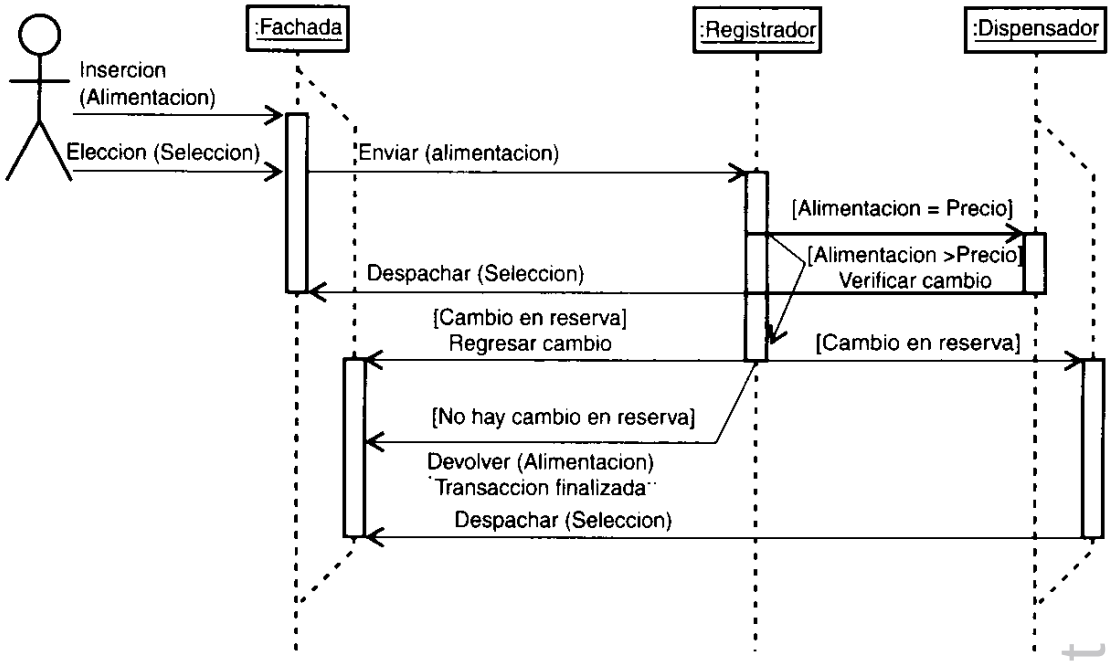
La figura 9.8 muestra un diagrama luego de agregar el escenario de "Monto incorrecto".

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 9.8**

*El diagrama de secuencias luego de agregar el escenario de "Monto incorrecto" al caso de uso "Comprar gaseosa".*




Ahora agregaremos el escenario "Sin gaseosa".

1. Una vez que el cliente elige una marca agotada, la máquina mostrará un mensaje de "Agotado".
2. La máquina mostrará un mensaje que solicitará al cliente que haga otra elección.
3. El cliente tendrá la opción de oprimir un botón para que se le regrese su dinero.
4. Si el cliente elige una marca en existencia, todo procederá como en el mejor escenario si el monto insertado es correcto. Si no lo es, la máquina seguirá por el escenario del "Monto incorrecto".
5. Si el cliente elige otra marca agotada, el proceso se repetirá hasta que el cliente elija una marca en existencia o presione un botón que le regrese su dinero.

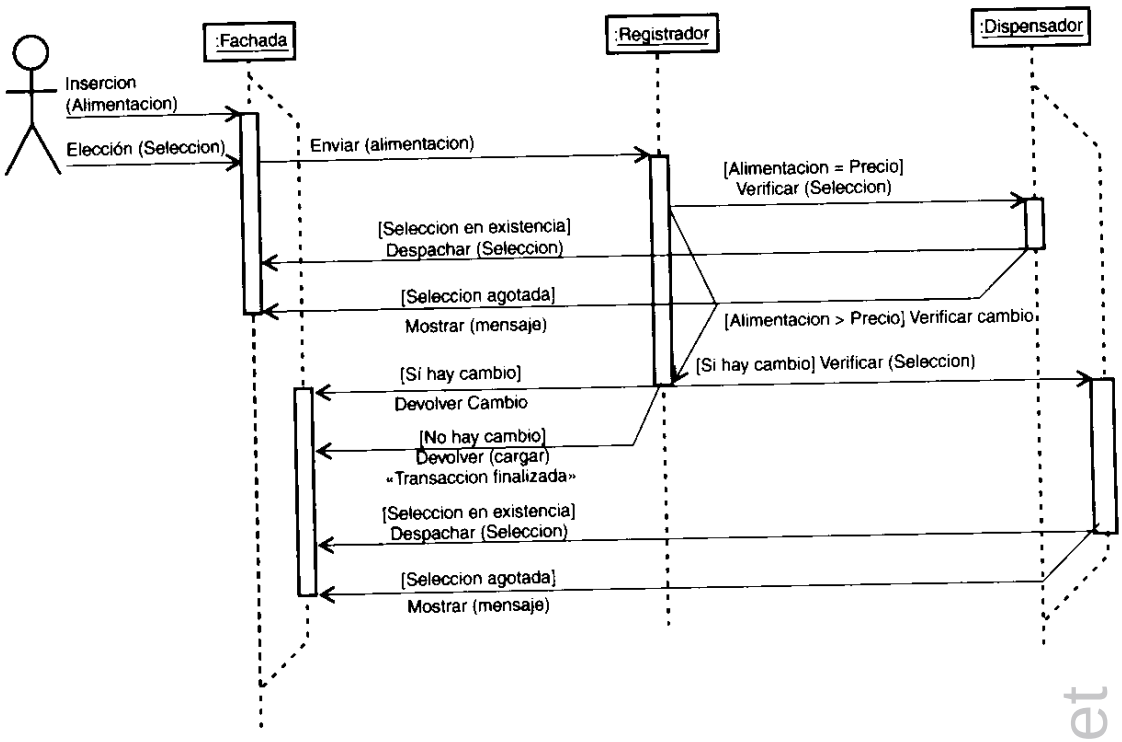
La figura 9.9 le muestra el diagrama de secuencias genérico de la máquina de gaseosas con los escenarios "Monto incorrecto" y "Sin marca".

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 9.9**

*El diagrama de secuencias genérico de la máquina de gaseosas luego de agregarle el escenario "Sin marca" a la figura 9.8.*




Si empieza a pensar que un diagrama de secuencias está implícito en cada caso de uso, ya tiene la idea.

# Creación de un objeto en la secuencia

En los ejemplos que hemos visto ha analizado distintos tipos de mensajes, diagramas de secuencias genérico y de instancias, así como estructuras de control. Otro concepto importante relacionado con los diagramas de secuencias, particularmente cuando diseñe software, es la creación de objetos.

Con frecuencia se da el caso de que un programa orientado a objetos debe crear un objeto. Recuerde que en términos del software, una clase es una plantilla para crear un objeto (como un molde de galletas para crear una galleta). ¿Cómo representaría la creación de un objeto cuando represente una secuencia de interacciones entre objetos?

El caso de uso "Crear una propuesta" del ejemplo de la LAN en una firma de consultoría nos muestra una instancia de la creación de objetos. En este ejemplo concebirá la LAN en el entendido de que todo se realiza mediante la red. Si damos por hecho que el consultor ya ha iniciado una sesión en la LAN, la secuencia que modelará quedará como sigue:

1. El consultor querrá volver a utilizar partes de una propuesta existente y busca en un área centralizada de la red una propuesta adecuada.
2. Si el consultor encuentra una propuesta adecuada, abrirá el archivo y, en el proceso, abrirá el software integrado para la oficina relacionada. El consultor guardará el archivo con un nuevo nombre, con lo que creará un archivo nuevo para la nueva propuesta.

www.elsolucionario.net

www.elsolucionario.net

3. Si el consultor no encuentra una propuesta, abrirá la aplicación de oficina y creará un archivo para la propuesta.
4. Al trabajar en la propuesta, el consultor utilizará las aplicaciones del software integrado para oficina.
5. Cuando el consultor finalice la propuesta, la guardará en el área de almacenamiento centralizada.

Además de la creación de objetos (en este caso, de archivos), esta secuencia trae consigo el uso de "si" así como de un ciclo "mientras".

TÉRMINO NUEVO

Primero veamos lo relacionado con la creación de objetos. Cuando una secuencia da por resultado la creación de un objeto, tal objeto se representará de la forma usual: como un rectángulo con nombre. La diferencia es que no lo colocará en la parte superior del diagrama de secuencias, sino que lo colocará junto con la dimensión vertical, de modo que su ubicación corresponda al momento en que se cree. El mensaje que creará al objeto se nombrará "Crear()". Los paréntesis implican una operación: en un lenguaje orientado a objetos, una operación *constructor* genera un objeto.

En lugar de usar "Crear()" o "Create()" para etiquetar la flecha de un mensaje de creación de un objeto, podría valerse de un estereotipo llamado «Crear».

En el caso de "mientras", a este control de flujo lo representará colocando la condición mientras ("mientras se trabaja en una propuesta") entre corchetes, con un asterisco (*) antes del primer corchete.

La figura 9.10 le muestra el diagrama de secuencias del caso de uso "Crear una propuesta".

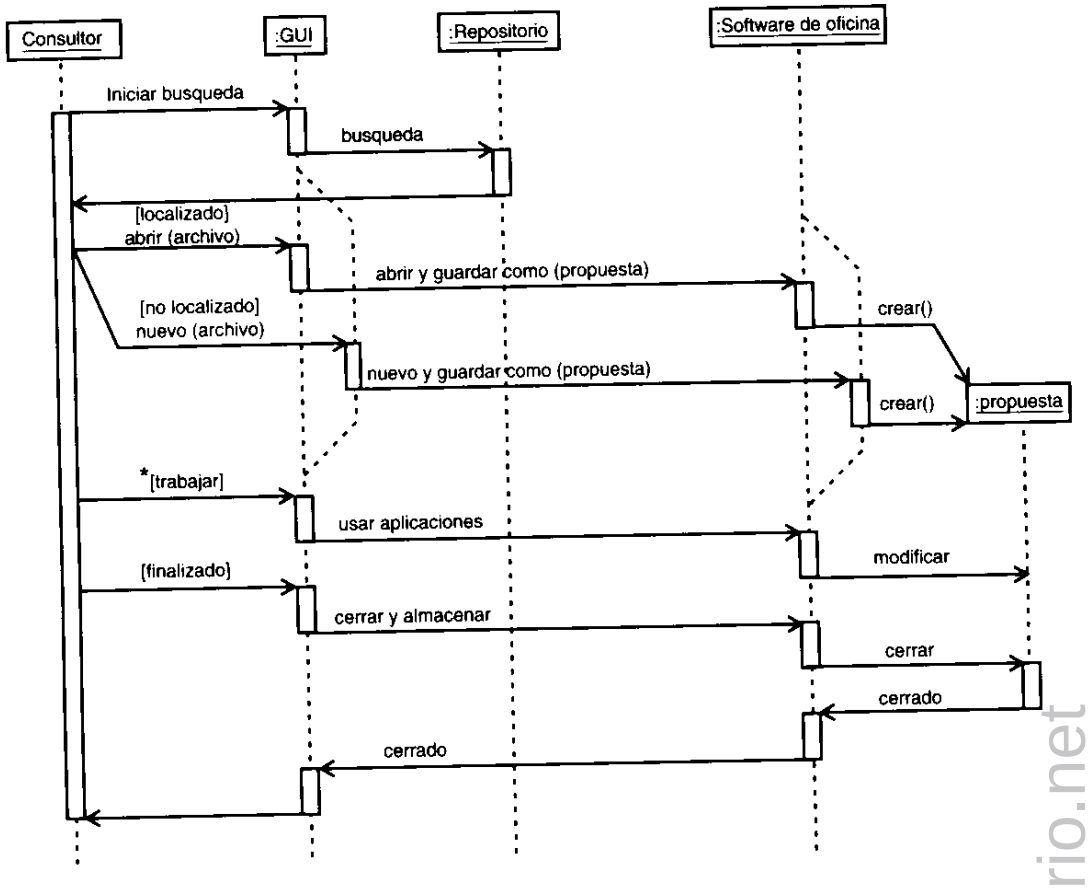
Este ejemplo representa una abstracción en la que he omitido detalles que no nos competen en lo particular. Esto lo he hecho de dos formas. Primero, obvié los detalles de la LAN, como lo mencioné. También vea que la GUI es un objeto en el diagrama de secuencias, y que no he incluido toda la complejidad del caso de uso "Teclazo" del ejemplo anterior. Los detalles de la interacción de la GUI con el sistema operativo, la CPU y el monitor no son importantes en este caso.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 9.10**

*El diagrama de secuencias del caso de uso "Crear una propuesta".*




# Cómo representar la recursividad

**TERMINO NUEVO**

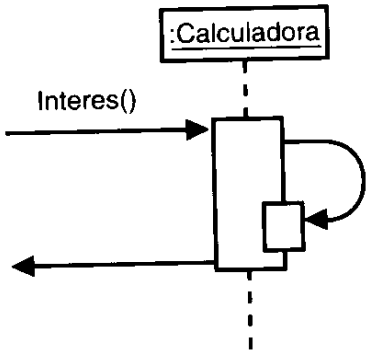
En ocasiones un objeto cuenta con una operación que se invoca a sí misma. A esto se le conoce como *recursividad*, y es una característica fundamental de varios lenguajes de programación.

He aquí un ejemplo. Suponga que uno de los objetos en su sistema sea una calculadora, y que una de sus operaciones sea el cálculo de intereses. Para calcular el interés compuesto para un periodo que incluya a varios periodos, la operación del cálculo de intereses del objeto tendrá que invocarse a sí misma varias veces.

Para representar esto en el UML, dibujará una flecha de mensaje fuera de la activación que signifique la operación, y un pequeño rectángulo sobrepuesto en la activación. Dibuje una flecha de modo que apunte al pequeño rectángulo, y una que regresa al objeto que inició la recursividad. La figura 9.11 muestra lo anterior.

**FIGURA 9.11**

*Cómo representar la recursividad en un diagrama de secuencias.*



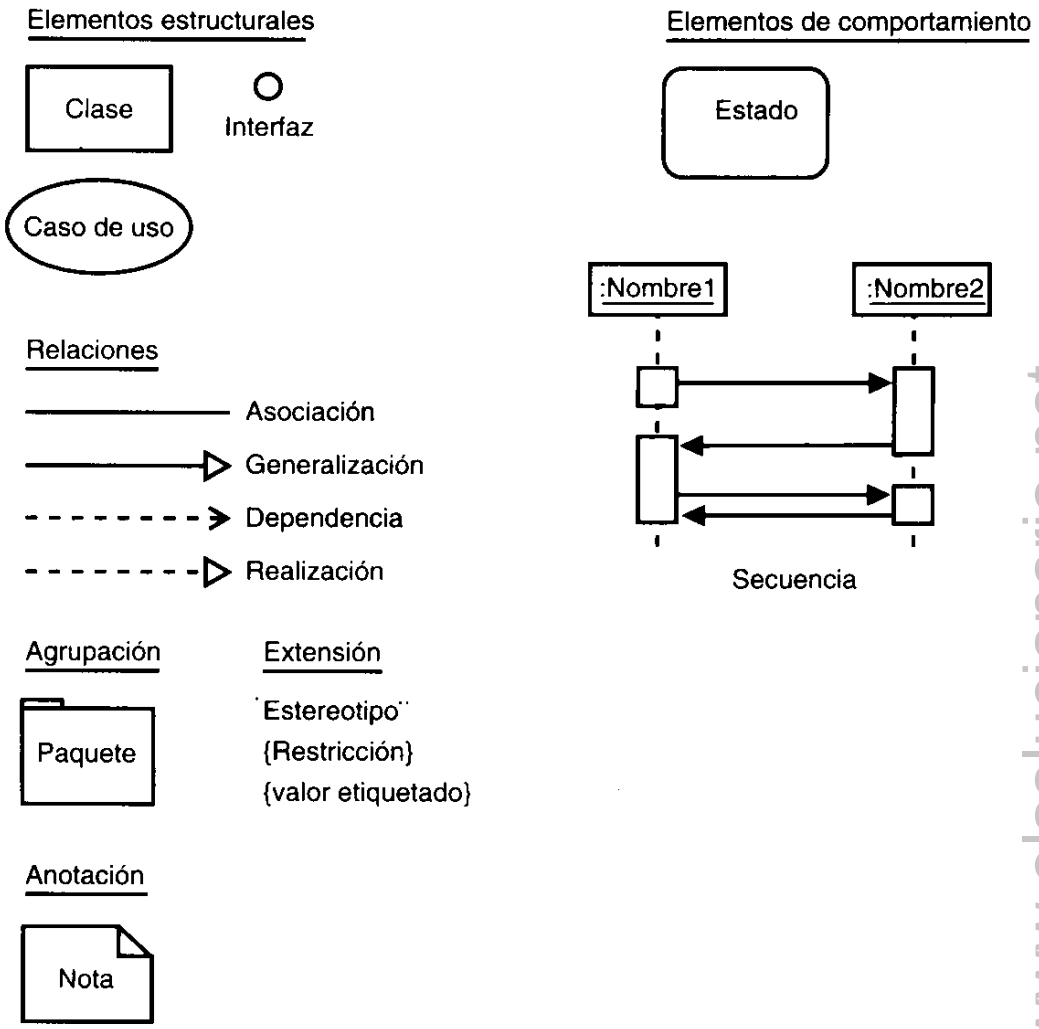

www.elsolucionario.net

# Adiciones al panorama

Ahora podrá agregar otro diagrama a su panorama del UML. Dado que se refiere al comportamiento de los objetos, el diagrama de secuencias iría bajo la categoría "Elementos de comportamiento". La figura 9.12 actualiza su creciente panorama.

**FIGURA 9.12**

*El panorama del UML con la adición del diagrama de secuencias.*




www.elsolucionario.net

# Resumen

El diagrama de secuencias UML agrega la dimensión del tiempo a las interactividades de los objetos. En el diagrama, los objetos se colocan en la parte superior y el tiempo avanza de arriba hacia abajo. La línea de vida de un objeto desciende de cada uno de ellos. Un pequeño rectángulo de la línea de vida de un objeto representa una *activación* (la ejecución de una de las operaciones del objeto). Puede incorporar los estados de un objeto colocándolos junto a su línea de vida.

Los mensajes (simples, sincrónicos y asincrónicos) son flechas que conectan a una línea de vida con otra. La ubicación del mensaje en la dimensión vertical representará el momento en que sucede dentro de la secuencia. Los mensajes que ocurren primero están más cerca de la parte superior del diagrama, y los que ocurren después cerca de la parte inferior.

www.elsolucionario.net

Un diagrama de secuencias puede mostrar ya sea una instancia (un escenario) de un caso de uso, o puede ser genérico e incorporar todos los escenarios de un caso de uso. Los diagramas de secuencias genéricos con frecuencia dan la oportunidad de representar instrucciones condicionales y ciclos "mientras". Bordee a cada condición con corchetes, y haga lo mismo en un ciclo "mientras" pero anteceda al corchete izquierdo con un asterisco.

Cuando una secuencia incluya la creación de un objeto, lo representará como un rectángulo de la forma acostumbrada. Su posición en la dimensión vertical representarán el momento en que se creó.

En ciertos sistemas, una operación puede invocarse a sí misma. A esto se le conoce como *recursividad*. Se representa con una flecha que sale de la activación hacia sí misma, y un pequeño rectángulo sobrepuesto a la activación.

# Preguntas y respuestas

**P El diagrama de secuencias parece que podría ser útil para más que tan sólo el análisis de sistemas. ¿Puedo usarlo para mostrar la interactividad en una empresa?**

**R** Así es. Los objetos pueden ser los actores principales, y los mensajes pueden ser simples transferencias de control.

**P Usted indicó la forma de representar la creación de objetos en un diagrama de secuencias. ¿Los objetos también se destruyen, y si es así, cómo lo represento?**

**R** Los objetos, en efecto, se destruyen. Podrá representar la destrucción de un objeto con una "X" al final de la línea de vida correspondiente a tal objeto.

# Taller

Ahora que ha vuelto hacia los objetos y ha visto sus interactividades, venga y responda algunas preguntas y realice algunos ejercicios para reafirmar su conocimiento de los diagramas de secuencias. Encontrará las respuestas en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. Defina mensaje *sincrónico* y mensaje *asincrónico*.
2. En un diagrama de secuencias genérico ¿cómo representaría el control de flujo implícito en una instrucción condicional?
3. ¿Cómo representaría el control de flujo implícito en una instrucción de ciclo "mientras"?
4. En un diagrama de secuencias ¿cómo representaría a un objeto recién creado?

www.elsolucionario.net

www.elsolucionario.net

## Ejercicios

1. Cree un diagrama de secuencias de instancias que muestre lo que ocurre cuando envía con éxito un fax. Esto es, modele las interactividades entre objetos en el mejor escenario del caso de uso "enviar fax" de una máquina de fax. Incluya los objetos de la máquina que envía, la que recibe, el fax y un "intercambio" central que encause a los faxes y a las llamadas telefónicas.
2. Cree un diagrama de secuencias genérico que incluya escenarios infructuosos (línea ocupada, error de la máquina que envía), así como del mejor escenario indicado en el ejercicio 1.

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 10

# Diagramas de colaboraciones

Ahora veremos lo correspondiente a un diagrama que es similar al que vimos en la hora anterior. Este también muestra la colaboración entre los objetos, pero de una forma significativamente diferente del diagrama de secuencias.

En esta hora se tratarán los siguientes temas:

- Qué es un diagrama de colaboraciones
- Cómo aplicar un diagrama de colaboraciones
- Uso de "si" y "mientras"
- Anidamiento
- Objetos activos y concurrencia
- Sincronización
- Dónde encajan los diagramas de colaboraciones en el UML

www.elsolucionario.net

www.elsolucionario.net

Los diagramas de colaboraciones muestran la forma en que los objetos colaboran entre sí, tal como sucede con un diagrama de secuencias. Muestran los objetos junto con los mensajes que se envían entre ellos. Si el diagrama de secuencias hace eso, ¿por qué el UML requeriría otro diagrama?, ¿qué no son lo mismo?, ¿no es una pérdida de tiempo?

Ambos tipos de diagrama son similares. De hecho, son *semánticamente equivalentes*. Esto significa que representan la misma información, y podrá convertir un diagrama de secuencias en un diagrama de colaboraciones equivalente y viceversa.

Como se infiere, es útil contar con ambas formas. Los diagramas de secuencias destacan la sucesión de las interacciones. Los diagramas de colaboraciones destacan el contexto y organización general de los objetos que interactúan. He aquí otra forma de encontrar la diferencia: el diagrama de secuencias se organiza de acuerdo al tiempo, y el de colaboración de acuerdo al espacio.

# Qué es un diagrama de colaboraciones

Un diagrama de objetos muestra a los objetos como tales y sus relaciones entre sí. Un diagrama de colaboraciones es una extensión de uno de objetos. Además de las relaciones entre objetos, el diagrama de colaboraciones muestra los mensajes que se envían los objetos entre sí. Por lo general, evitará la multiplicidad dado que podría ser fuente de confusión.

Para representar un mensaje, dibujará una flecha cerca de la línea de asociación entre dos objetos, esta flecha apunta al objeto receptor. El tipo de mensaje se mostrará en una etiqueta cerca de la flecha; por lo general, el mensaje le indicará al objeto receptor que ejecute una de sus operaciones. El mensaje finalizará con un par de paréntesis, dentro de los cuales colocará los parámetros (en caso de haber alguno) con los que funcionará la operación.

Mencioné que podrá convertir cualquier diagrama de secuencias en diagrama de colaboraciones y viceversa. Por medio de esto podrá representar la información de secuencia en un diagrama de colaboraciones. Para ello, agregará una cifra a la etiqueta de un mensaje, misma que corresponderá a la secuencia propia del mensaje. La cifra y el mensaje se separan mediante dos puntos (:).

La figura 10.1 le muestra la simbología del diagrama de colaboraciones.

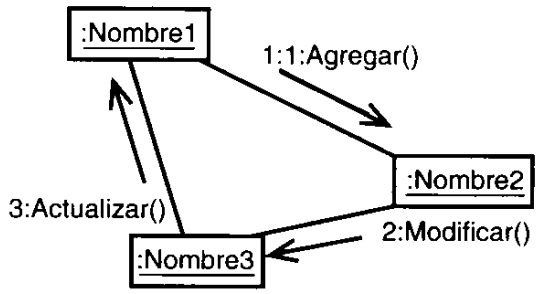
Aprovechemos la equivalencia de ambos tipos de diagramas. Para desarrollar los conceptos de los diagramas de colaboraciones volveremos a ver los ejemplos que revisamos la hora anterior. Conforme lo haga, verá más conceptos.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 10.1**

*Simbología del diagrama de colaboraciones.*




# La GUI

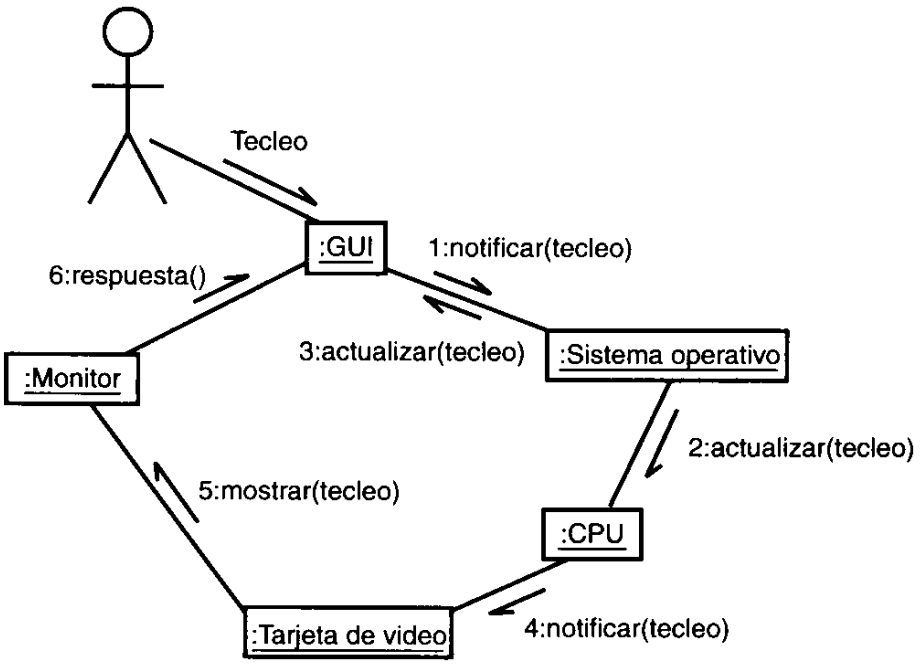
Este ejemplo es el caso más directo. Un actor inicia la secuencia de interacción al oprimir una tecla, con lo que los mensajes ocurrirán de manera secuencial. Tal secuencia (a partir de la hora anterior) es:

1. La GUI notifica al sistema operativo que se oprimió una tecla.
2. El sistema operativo le notifica a la CPU.
3. El sistema operativo actualiza la GUI.
4. La CPU notifica a la tarjeta de vídeo.
5. La tarjeta de vídeo envía un mensaje al monitor.
6. El monitor presenta el carácter alfanumérico en la pantalla, con lo que se hará evidente al usuario.

La figura 10.2 muestra la forma de representar esta secuencia de interacciones en un diagrama de colaboraciones. El diagrama muestra la figura agregada que representa al usuario que inicia la secuencia, aunque esta figura no es parte de la simbología de este diagrama.

**FIGURA 10.2**

*Un diagrama de colaboraciones para el ejemplo de la GUI.*




www.elsolucionario.net

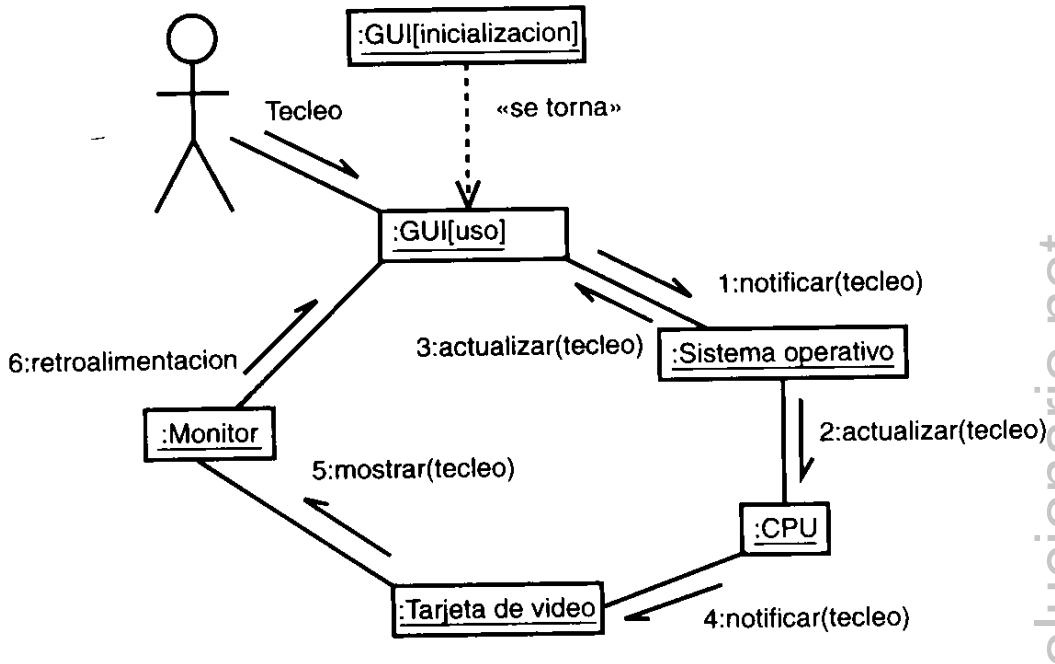
## Cambios de estado

Puede mostrar los cambios de estado en un objeto en un diagrama de colaboraciones. En el rectángulo del objeto indique su estado. Agregue otro rectángulo al diagrama que haga las veces del objeto e indique el estado modificado. Conecte a los dos con una línea discontinua y etiquete la línea con un estereotipo «se torna».

La figura 10.3 ilustra un cambio de estado para la GUI, que muestra que el estado de inicialización se convierte en el estado operativo.

**FIGURA 10.3**

*Un diagrama de colaboraciones puede incorporar cambios de estado.*




# La máquina de gaseosas

Las cosas se hacen más interesantes cuando aplica las condiciones a una situación real, como lo hizo en la hora anterior con la máquina de gaseosas. Iniciemos con la mejor situación del caso de uso "Comprar gaseosa", donde la secuencia es:

1. El cliente inserta el dinero en la alcancía que se encuentra en la fachada de la máquina.
2. El cliente hace su elección.
3. El dinero viaja hacia el registrador.
4. El registrador verifica si la gaseosa elegida está en el dispensador.
5. Dado que es la mejor situación, asumimos que sí hay gaseosas, y el registrador actualiza su reserva de efectivo.
6. El registrador hace que el dispensador entregue la gaseosa en la fachada de la máquina.

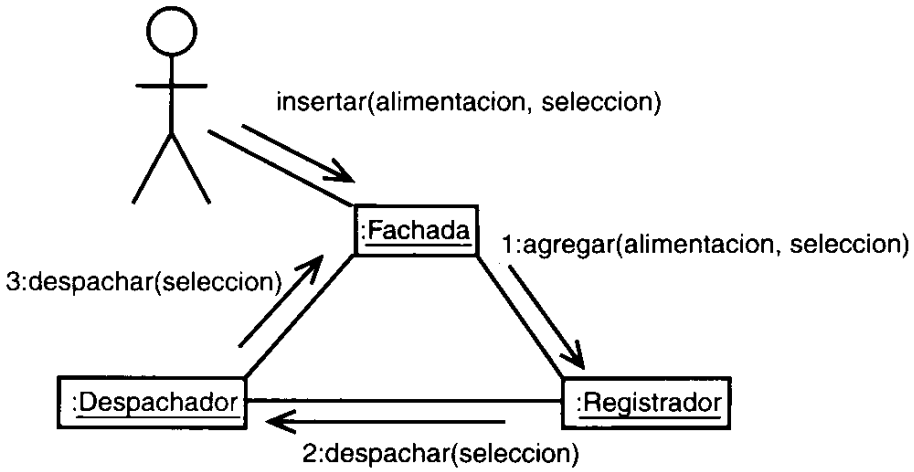
El diagrama de colaboraciones es directo, como lo muestra la figura 10.4.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 10.4**

*El diagrama de colaboraciones para el mejor caso de "Comprar gaseosa".*




Ahora, agreguemos el caso de "cantidad incorrecta de dinero". El diagrama tiene que contabilizar varias condiciones:

1. El usuario ha introducido más dinero que el necesario para la compra.
2. La máquina cuenta con la cantidad adecuada de cambio.
3. La máquina no tiene la cantidad correcta de cambio.

Usted representará las condiciones de la misma forma en que las representó en el diagrama de secuencias. Colocará la condición entre corchetes, misma que se antecede a la etiqueta. Lo importante es coordinar las condiciones con la numeración.

Esto podría ser algo complicado, por lo que haremos el diagrama por secciones. Empezaremos con la condición donde el usuario ha insertado más dinero del indicado en el precio y el registrador cuenta con el cambio adecuado. Agregará el paso de la máquina al devolver el cambio al cliente, y agregará las condiciones entre corchetes. El paso que devuelve el cambio es una consecuencia del que verifica si hay cambio. Para indicar esto en el paso de devolver cambio utilizará el mismo número del mensaje que verifica el cambio, y agregará un punto decimal y un uno. A esto se le conoce como *anidación*. La figura 10.5 le muestra los detalles.

¿Qué ocurre cuando la máquina no cuenta con el cambio correcto? Tendrá que mostrar un mensaje que lo indique, devuelva el dinero y pida al usuario que inserte el importe correcto. Así, la transacción habrá finalizado.

Cuando agregue esta condición, agregará una bifurcación en el control de flujo. Numerará esta bifurcación como un mensaje anidado. Dado que es el segundo mensaje anidado, habrá un 2 luego del punto decimal. Finalmente, y debido a que la transacción habrá finalizado, hará clara esta situación mediante la adición de un estereotipo "transacción finalizada" en este mensaje, y otro en el mensaje que despacha la gaseosa. La figura 10.6 presentará la situación.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 10.5**

*El diagrama de colaboraciones con parte de la situación "monto de dinero inadecuado".*

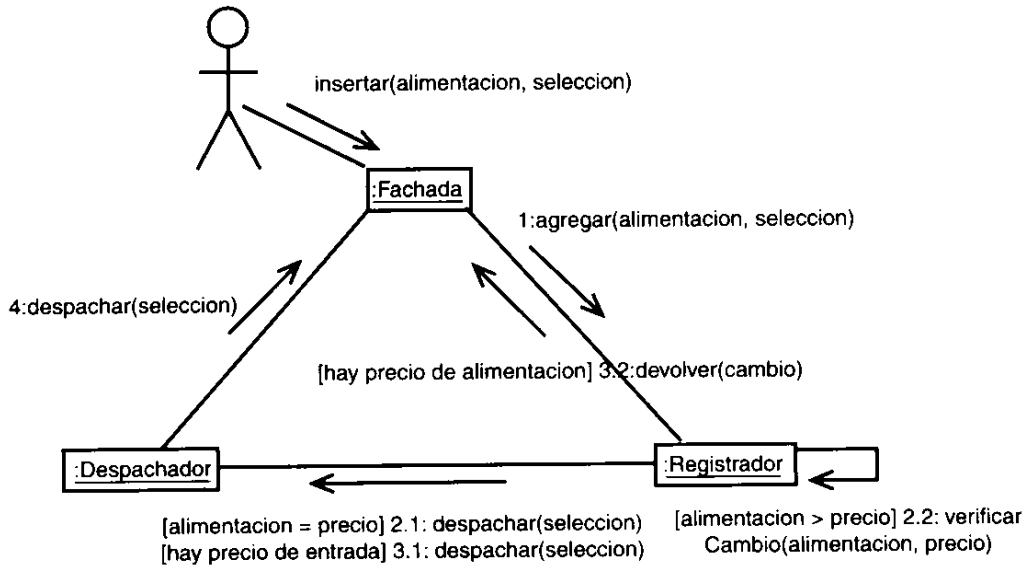
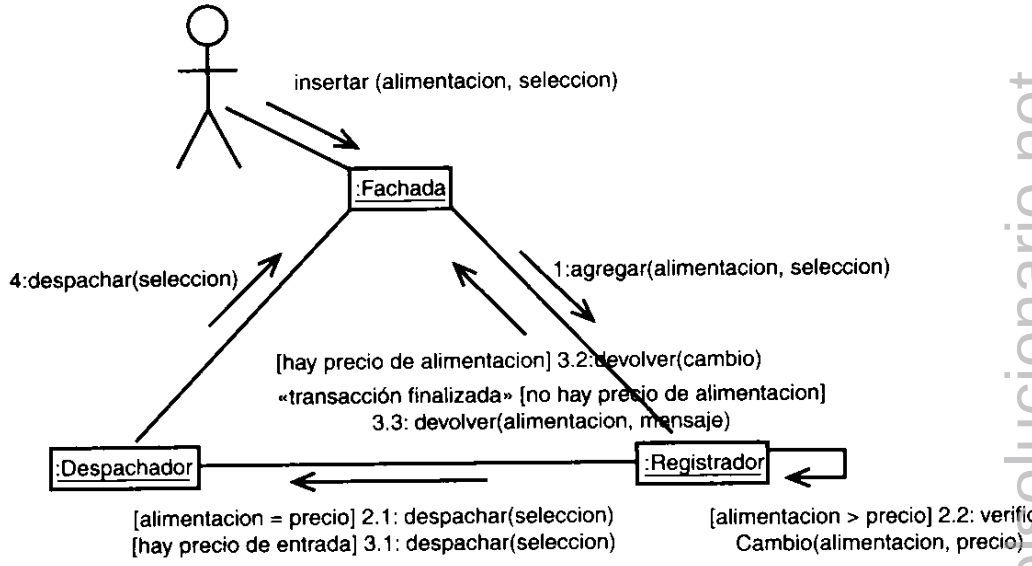



**FIGURA 10.6**

*El diagrama de colaboraciones "Comprar gaseosa" con toda la situación "monto de dinero inadecuado".*




En el taller, al finalizar esta hora, habrá un ejercicio que le pedirá que complete el diagrama de colaboraciones mediante la adición de la situación "no hay gaseosa".

# Creación de un objeto

Para mostrar la creación de objetos, volveré al caso de uso "Crear propuesta" de la firma de consultoría. Una vez más, la secuencia que modelará será:

1. El consultor buscará en el área de almacenamiento centralizada de la red una propuesta adecuada en la cual basarse.
2. Si el consultor localiza una propuesta adecuada, la abrirá y en el proceso abrirá la aplicación de oficina. El consultor guardará el archivo bajo un nuevo nombre, con lo que creará un nuevo archivo para la nueva propuesta.
3. Si el consultor no encuentra una propuesta, abrirá la aplicación de oficina y generará un nuevo archivo.

www.elsolucionario.net

www.elsolucionario.net

4. Al trabajar en la propuesta, el consultor utilizará los componentes de la aplicación de oficina.
5. Cuando el usuario finalice la propuesta, la guardará en el área de almacenamiento centralizada.

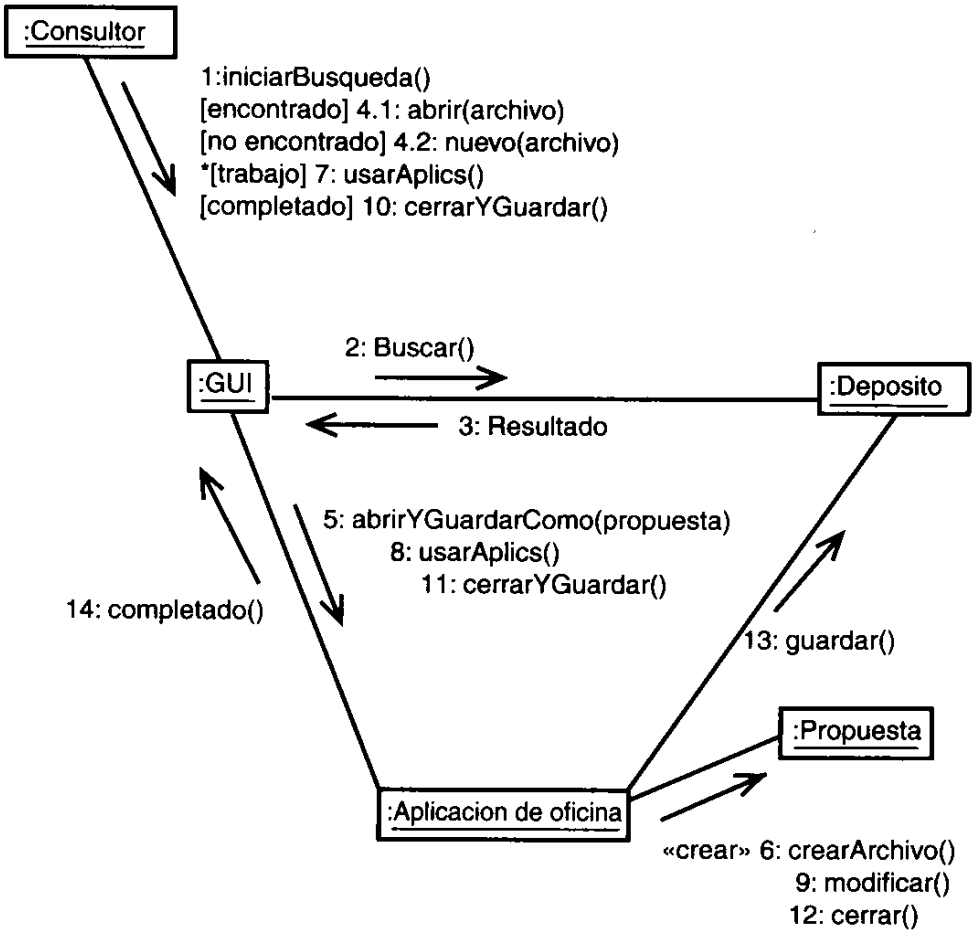
Para mostrar la creación de un objeto, agregará un estereotipo "crear" al mensaje que genera al objeto.

Una vez más, utilizará instrucciones "si" (if) y mensaje anidados. También trabajará con un ciclo "mientras" (while). Como en el diagrama de secuencias, para representar a "mientras", colocará esta condición entre corchetes y antecederá al del lado izquierdo con un asterisco.

La figura 10.7 le muestra este diagrama de colaboraciones completo con la creación del objeto y "mientras".

**FIGURA 10.7**

*El diagrama de colaboraciones "Crear una propuesta".*




# Algunos conceptos más

Aunque ha visto algunas bases, no ha visto todo lo relacionado con los diagramas de colaboraciones. Los conceptos en esta sección son un poco esotéricos, pero podrían serle útiles en sus esfuerzos para analizar sistemas.

www.elsolucionario.net

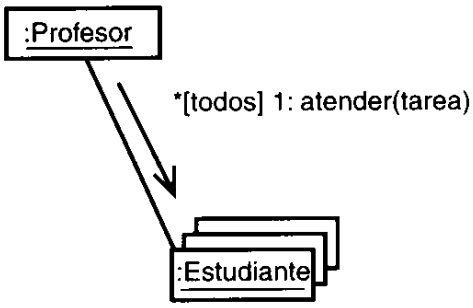
www.elsolucionario.net

## Varios objetos receptores en una clase

En ocasiones un objeto envía un mensaje a diversos objetos de la misma clase. Por ejemplo: un profesor le pide a un grupo de estudiantes que entreguen una tarea. En el diagrama de colaboraciones, la representación de los diversos objetos es una pila de rectángulos que se extienden "desde atrás". Agregará una condición entre corchetes precedida por un asterisco para indicar que el mensaje irá a todos los objetos. La figura 10.8 le muestra los detalles.

**FIGURA 10.8**

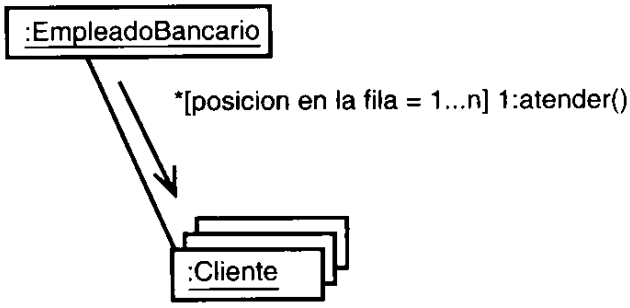
*Un objeto que envía un mensaje a diversos objetos de una clase.*




En algunos casos, el orden del mensaje enviado es importante. Por ejemplo, un empleado bancario dará servicio a cada cliente conforme fue llegando a la fila. Esto lo representará con un "mientras" cuya condición implicará orden (como en "posición en la fila = 1 ... n") junto con el mensaje y la pila de rectángulos (vea la figura 10.9).

**FIGURA 10.9**

*Un objeto que envía un mensaje a varios otros en un orden específico.*




## Representación de los resultados

Un mensaje podría ser una petición a un objeto para que realice un cálculo y devuelva un valor. Un objeto Cliente podría solicitar a un objeto Calculadora que calcule el precio total que sea la suma del precio de un elemento y el impuesto.

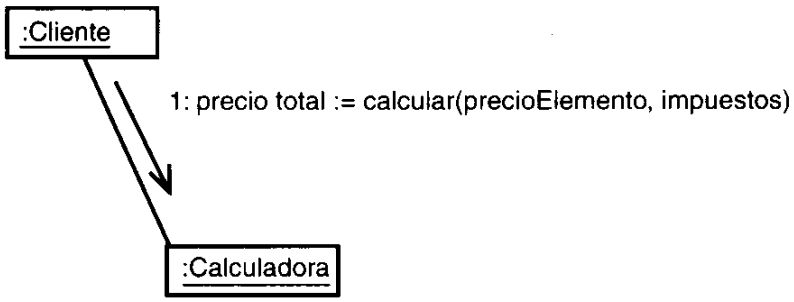
El UML le da una sintaxis para representar esta situación. Deberá escribir una expresión que tenga el nombre del valor devuelto a la izquierda, seguido de ":=", a continuación el nombre de la operación y las cantidades con que opera para producir el resultado. En este ejemplo, la expresión podría ser precioTotal := calcular(precioElemento, impuesto). La figura 10.10 le muestra la sintaxis de un diagrama de colaboraciones.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 10.10**

*Un diagrama de colaboraciones que incluye la sintaxis de un resultado.*



**TÉRMINO NUEVO** A la parte que está a la derecha de la expresión se le conoce como *firma del mensaje*.

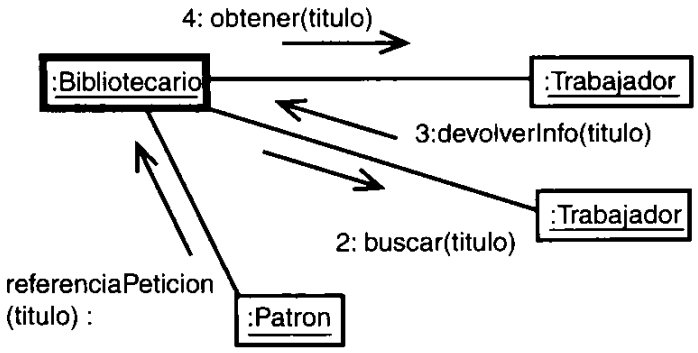
## Objetos activos

**TÉRMINO NUEVO** En algunas interacciones, un objeto específico controla el flujo. Este *objeto activo* puede enviar mensajes a los objetos pasivos e interactuar con otros objetos activos. En una biblioteca, un bibliotecario relaciona las peticiones a partir de un patrón, verifica la información de referencia en una base de datos, devuelve una respuesta al peticionario, asigna personas para reabastecer los libros, entre otras cosas. Un bibliotecario también interactúa con otros que realicen las mismas operaciones. Al proceso de que dos o más objetos activos hagan sus tareas al mismo tiempo, se le conoce como *concurrencia*.

El diagrama de colaboraciones representa a un objeto activo de la misma manera que a cualquier otro objeto, excepto que su borde será grueso y más oscuro. (Vea la figura 10.11.)

**FIGURA 10.11**

*Un objeto activo controla el flujo en una secuencia. Se representa como un rectángulo con un borde grueso en negro.*




## Sincronización

Otro caso con el que se puede encontrar es que un objeto sólo puede enviar un mensaje después de que otros mensajes han sido enviados. Es decir, el objeto debe "sincronizar" todos los mensajes en el orden debido.

www.elsolucionario.net

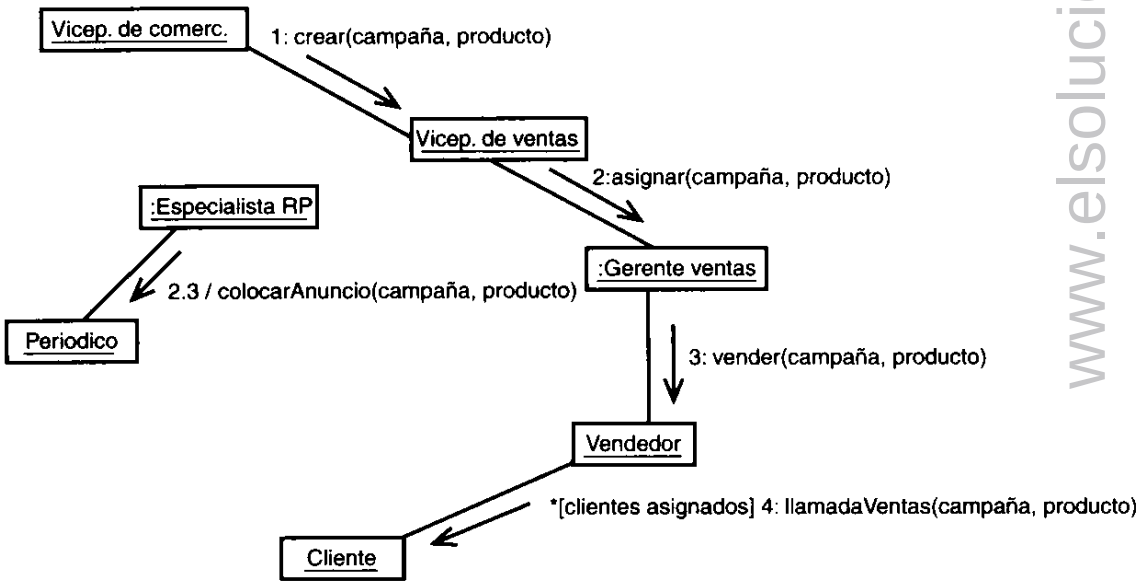
Un ejemplo aclarará esto. Suponga que sus objetos son personas en un corporativo, y que están ocupados en la campaña de un nuevo producto. He aquí la secuencia de interacciones:

1. El vicepresidente de comercialización le pide al de ventas que cree una campaña para un producto en particular.
2. El vicepresidente de ventas crea la campaña y la asigna al gerente de ventas.
3. El gerente de ventas instruye a un agente de ventas para que venda el producto de acuerdo con la campaña.
4. El agente de ventas hace llamadas para vender el producto a los clientes en potencia.
5. Luego de que el vicepresidente de ventas ha dado la comisión y el gerente de ventas ha expedido la directiva (esto es, cuando se han completado los pasos 2 y 3), un especialista en relaciones públicas de la corporación hará una llamada al periódico local y colocará un anuncio de la campaña.

¿Cómo representará la posición del paso cinco en la secuencia? Nuevamente, el UML le da una sintaxis. En lugar de anteceder este mensaje con una etiqueta numérica, lo antecederá con una lista de mensajes que tendrán que completarse antes de que se realice el paso cinco. La lista de elementos se separará mediante una coma, y finalizará con una diagonal. La figura 10.12 le muestra el diagrama de colaboraciones en este ejemplo.

**FIGURA 10.12**

*La sincronización de mensajes en un diagrama de colaboraciones.*




# Adiciones al panorama

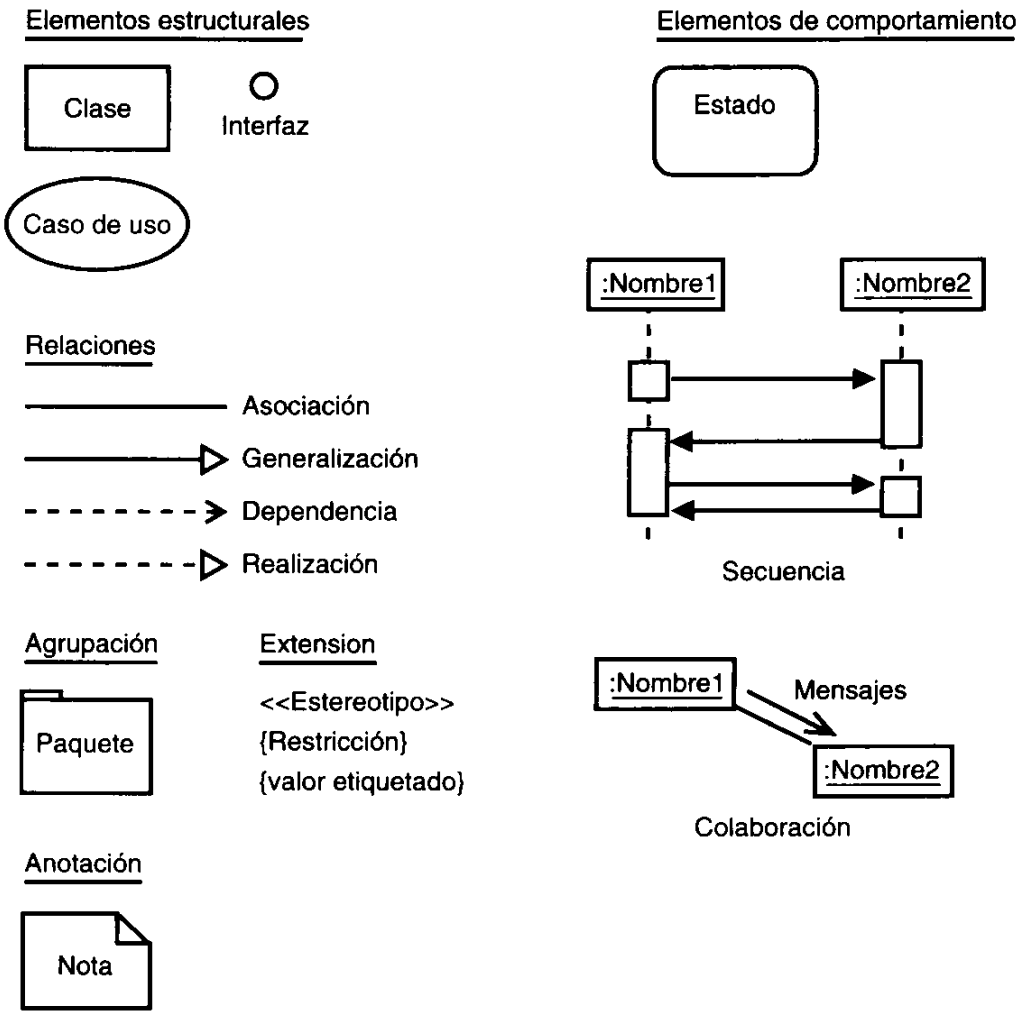
Ahora podrá agregar el diagrama de colaboraciones a su panorama del UML. Es otro elemento de comportamiento, como se aprecia en la figura 10.13.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 10.13**

*El panorama del UML, que incluye al diagrama de colaboraciones.*




# Resumen

Un diagrama de colaboraciones es otra forma de presentar la información en un diagrama de secuencias. Ambos tipos de diagramas son semánticamente equivalentes y se recomienda usar ambos cuando construya el modelo de un sistema. El diagrama de secuencias se organiza de acuerdo al tiempo, y el de colaboración de acuerdo al espacio.

El diagrama de colaboraciones muestra las asociaciones entre objetos, así como los mensajes que pasan de un objeto a otro. El mensaje se representa con una flecha junto a la línea de asociación, y una etiqueta numerada que muestra el contenido del mensaje. El número representa el turno del mensaje en la secuencia.

Las condicionales se representan como antes, mediante la colocación de la instrucción condicional entre corchetes. Para representar un ciclo "mientras", anteceda al corchete izquierdo con un asterisco.

Algunos mensajes provienen de otros. El esquema de numeración de las etiquetas representa esto de forma muy similar a los manuales técnicos que muestran sus encabezados y subtítulos: con un sistema de numeración que utiliza puntos decimales para representar los niveles del anidamiento.

www.elsolucionario.net

www.elsolucionario.net

Los diagramas de colaboraciones le permiten modelar varios objetos receptores en una clase, ya sea que los objetos reciban o no los mensajes en un orden específico. También podrá representar objetos activos que controlen el flujo de los mensajes, así como los mensajes que se sincronizan con otros.

# Preguntas y respuestas

**P ¿Realmente tengo que incluir a ambos diagramas (el de colaboraciones y el de secuencias) en la mayoría de los modelos UML que genere?**

**R** Se recomienda hacerlo. Ambos tipos de diagramas podrán estimular diversas ideas de los procesos durante el segmento de análisis en el proceso de desarrollo. El diagrama de colaboraciones clarifica las relaciones entre los objetos debido a que incluye los vínculos entre ellos. El de secuencia se enfoca en la secuencia de las interacciones. A su vez, la organización de su cliente podría incluir personas cuya idea de los procesos podría diferir entre ellos. Cuando tenga que presentar su modelo, un tipo de diagrama podría comprenderse mejor para ciertas personas.

# Taller

Ahora que ha comprendido los diagramas de secuencias y a sus hermanos, los de colaboración, pruebe y fortalezca su conocimiento con el cuestionario y los ejercicios. Como siempre, verá las respuestas en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿Cómo representa a un mensaje en un diagrama de colaboraciones?
2. ¿Cómo mostraría información secuencial en un diagrama de colaboraciones?
3. ¿Cómo mostraría los cambios de estado?
4. ¿Qué se entiende por la "equivalencia semántica" de dos tipos de diagramas?

## Ejercicios

1. En el ejemplo de la máquina de gaseosas, sólo mostré un diagrama de colaboraciones equivalente al diagrama de secuencias de instancia de la situación "importe incorrecto". Cree un diagrama de colaboraciones que corresponda al diagrama de secuencias genérico de la hora 9 para el caso de uso "Comprar gaseosa". Esto es, agregue la situación "Gaseosa agotada" al diagrama de colaboraciones de la figura 10.5.

www.elsolucionario.net

www.elsolucionario.net

2. En el diagrama de colaboraciones del caso de uso "Crear propuesta", el consultor busca en el área central de almacenamiento una propuesta adecuada para volverla a utilizar. Imagine a "buscar" como un mensaje enviado para buscar en una secuencia de archivos, y utilice las técnicas de modelado de la sección "Algunos conceptos más" para cambiar el diagrama de colaboraciones en la figura 10.6.

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 11

# Diagramas de actividades

Ahora veremos un tipo de diagrama que podría parecerle familiar, este diagrama le muestra los pasos en una operación o proceso.

En esta hora se tratarán los siguientes temas:

- Qué es un diagrama de actividades
- Aplicación de los diagramas de actividades
- Marcos de responsabilidad
- Adiciones al panorama

Si alguna vez ha tomado algún curso básico de programación, ya conocerá los diagramas de flujo. Siendo uno de los primeros modelos visuales que se aplicaron a la computación, el diagrama de flujo muestra una secuencia de pasos, procesos, puntos de decisión y bifurcaciones. A los programadores novatos se les invita a que utilicen este diagrama para conceptualizar problemas y derivar sus soluciones. La idea es convertir al diagrama de flujo

www.elsolucionario.net

www.elsolucionario.net

en la base del código. Con sus diversas características y tipos de diagramas, el UML es en cierta medida un diagrama de flujo con esteroides.

El diagrama de actividades del UML, tema de esta hora, es muy parecido a los viejos diagramas de flujo. Le muestra los pasos (conocidos como *actividades*) así como puntos de decisión y bifurcaciones. Es útil para mostrar lo que ocurre en un proceso de negocios u operación. Los encontrará como parte integral del análisis de un sistema.

# Qué es un diagrama de actividades

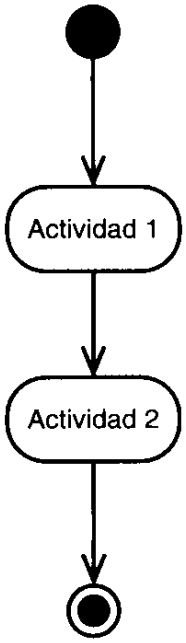
Para empezar, un diagrama de actividades ha sido diseñado para mostrar una visión simplificada de lo que ocurre durante una operación o proceso. Es una extensión de un diagrama de estados, mismo que ya conoció. El diagrama de estados muestra los estados de un objeto y representa las actividades como flechas que conectan a los estados. El diagrama de actividades resalta, precisamente, a las actividades.

A cada actividad se le representa por un rectángulo con las esquinas redondeadas (más angosto y ovalado que la representación del estado). El procesamiento dentro de una actividad se lleva a cabo y, al realizarse, se continúa con la siguiente actividad. Una flecha representa la transición de una a otra actividad. Al igual que el diagrama de estados, el de actividad cuenta con un punto inicial (representado por un círculo relleno) y uno final (representado por una diana).

La figura 11.1 le muestra el punto inicial y final, así como dos actividades y una transición.

**FIGURA 11.1**

*Transición de una actividad a otra.*




www.elsolucionario.net

www.elsolucionario.net

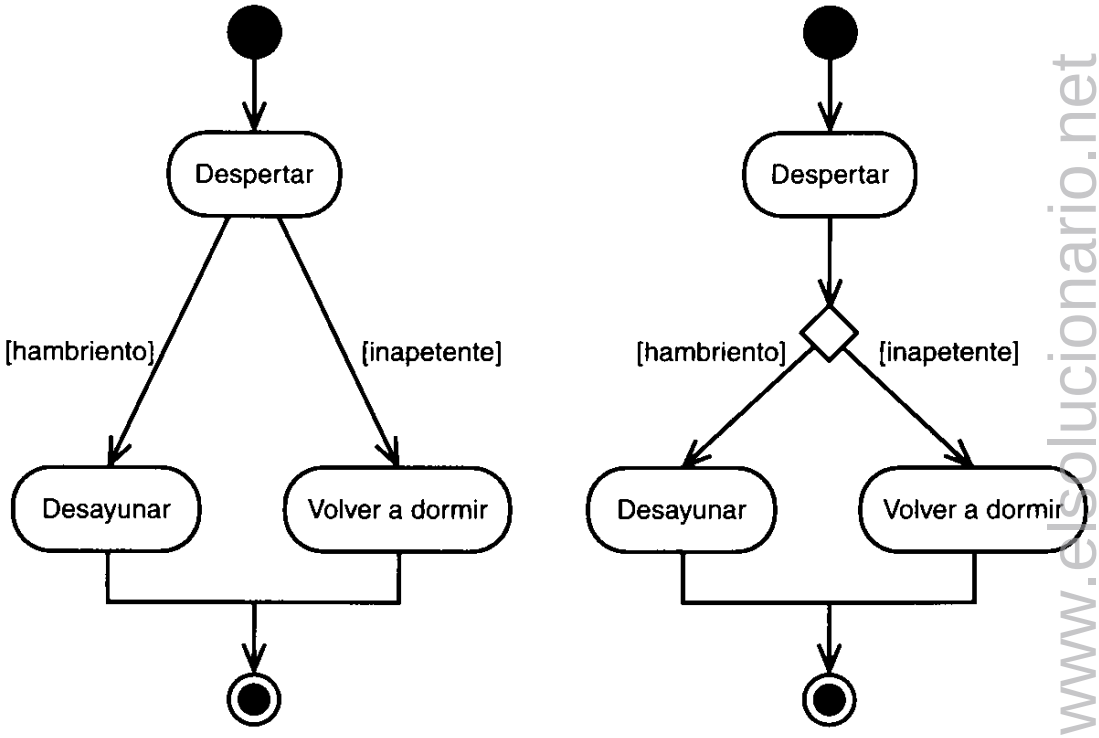
## Decisiones, decisiones, decisiones

Casi siempre una secuencia de actividades llegará a un punto donde se realizará alguna decisión. Ciertas condiciones le llevarán por un camino y otras por otro (pero ambas son mutuamente exclusivas).

Podrá representar un punto de decisión de una de dos formas: la primera es mostrar las rutas posibles que parten directamente de una actividad y la segunda es llevar la transición hacia un rombo —reminiscencias del símbolo de decisión en un diagrama de flujo— y que de allí salgan las rutas de decisión (como usuario de los antiguos diagramas de flujo, prefiero la segunda opción). De cualquier forma, indicará la condición con una instrucción entre corchetes junto a la ruta correspondiente. La figura 11.2 le muestra las posibilidades.

**FIGURA 11.2**

*Dos formas de mostrar una decisión.*




www.elsolucionario.net

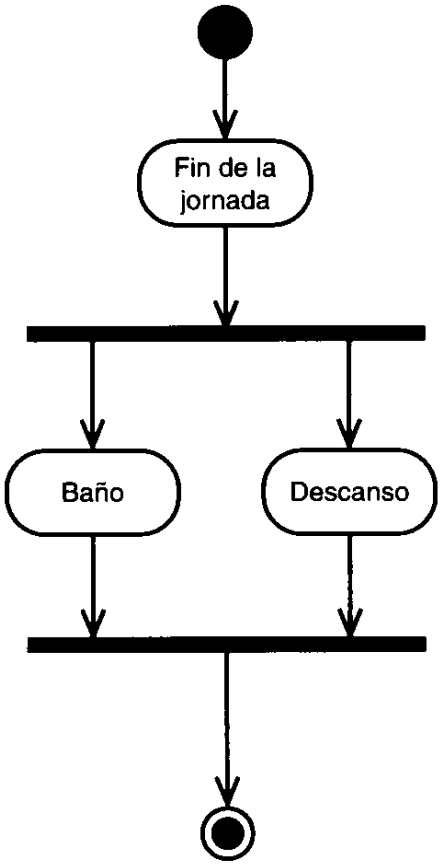
## Rutas concurrentes

Conforme modele actividades tendrá la oportunidad de separar una transición en dos rutas que se ejecuten al mismo tiempo (es decir, de forma concurrente) y luego se reúnan. Para representar esta división, utilizará una línea gruesa perpendicular a la transición y las rutas partirán de ella. Para representar la reincorporación, ambas rutas apuntarán a otra línea gruesa (vea la figura 11.3).

www.elsolucionario.net

**FIGURA 11.3**

*Representación de una transición que se bifurca en dos rutas que se ejecutan de forma concurrente y, luego, se reincorporan.*




## Indicaciones

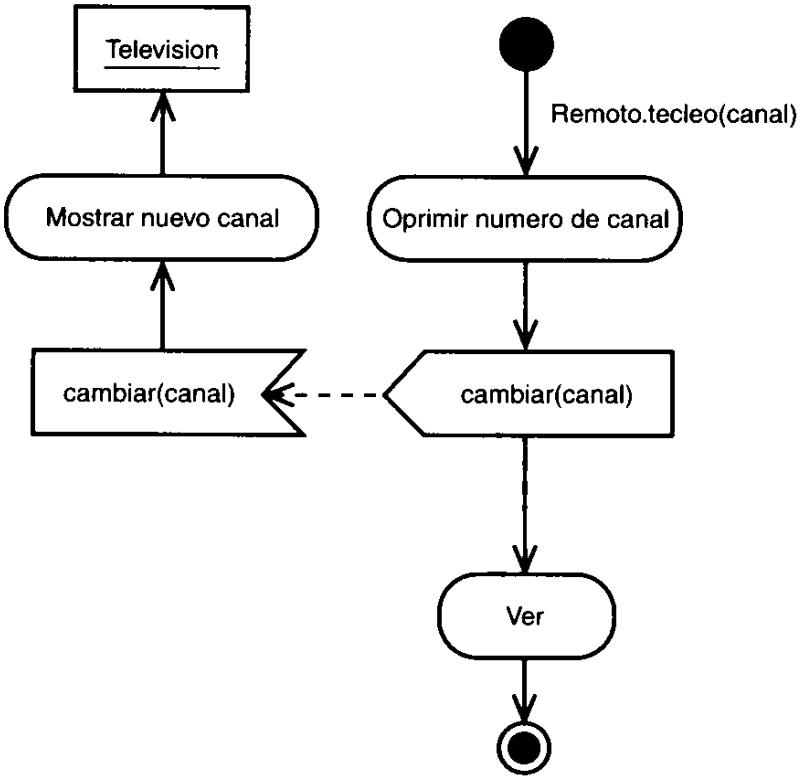
Durante una secuencia de actividades, es posible enviar una indicación. Cuando se reciba, la indicación provocará que se ejecute una actividad. El símbolo para enviar una indicación es un pentágono convexo, y el que la recibe es un pentágono cóncavo. La figura 11.4 le ayudará a clarificar la idea.

En términos del UML el pentágono convexo simboliza al *envío de un evento*; el cóncavo simboliza la *recepción del evento*.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 11.4**
*Envío y recepción de una indicación.*




# Aplicación de los diagramas de actividades

Veamos algunos ejemplos. Para empezar, diagramará una operación y posteriormente un proceso.

## Una operación: Fibs

¿Había visto la siguiente serie de números? 1, 1, 2, 3, 5, 8, 13,.... se conoce como la "serie de Fibonacci", pues este matemático medieval la escribió hace unos 800 años. Cada número es un "fib", así que el primer fib [o fib(1)] es 1, fib(2) es 1, fib(3) es 2, y así sucesivamente. La regla de cada fib, excepto en los dos primeros, es la suma del anterior par de fibs, por ejemplo, fib(8) tendría un valor de 21.

Suponga que una de sus clases es una calculadora, y que una de sus operaciones fuese la de calcular el enésimo fib y mostrarlo. A esta operación podría llamarla *calcularFib(n)*. Creemos un diagrama de actividades que modele a esta operación.

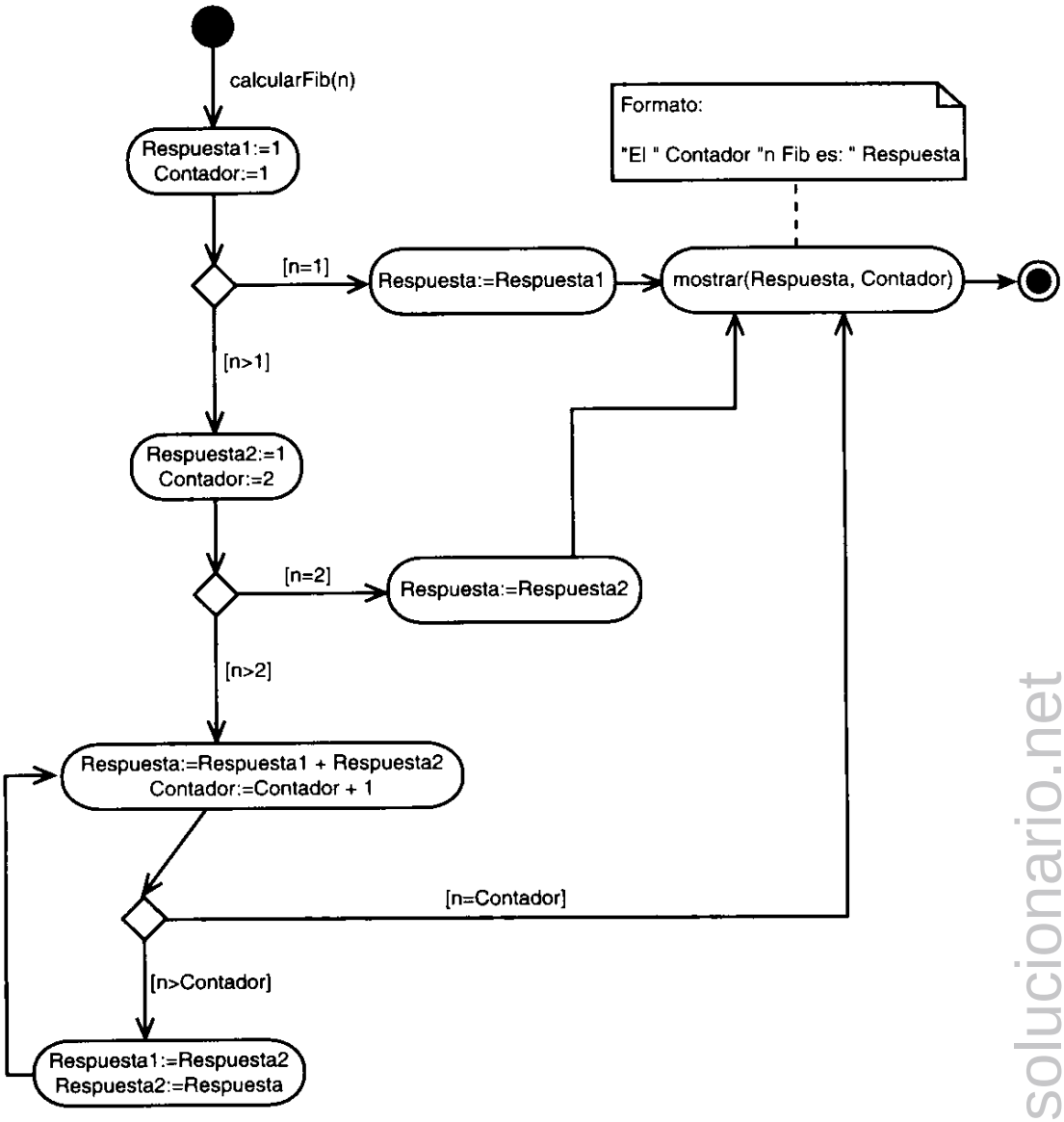
Requerirá algunas variables como son: un contador para llevar un control para verificar si se ha llegado al enésimo fib, una variable para conglomerar los resultados, y dos más para almacenar dos fibs que tendrá que sumar entre sí. La figura 11.5 le muestra el diagrama de actividades que realizaría la tarea.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 11.5**

*Un diagrama de actividades para calcularFib(n), una operación que calcula el enésimo número de Fibonacci.*




## Proceso de creación de un documento

Ahora volvamos nuestra atención de una operación a un proceso. Imagine las actividades necesarias para utilizar una aplicación de oficina para crear un documento. Una posible secuencia podría ser:

1. Abrir la aplicación para procesamiento de textos.
2. Crear un archivo.
3. Guardar el archivo con un nombre único en una carpeta.
4. Teclear el documento.
5. Si se necesitan ilustraciones, se abre la aplicación relacionada, se generan los gráficos y se colocan en el documento.
6. Si se necesita una hoja de cálculo, se abre la aplicación relacionada, se crea la hoja correspondiente y se coloca en el documento.

www.elsolucionario.net

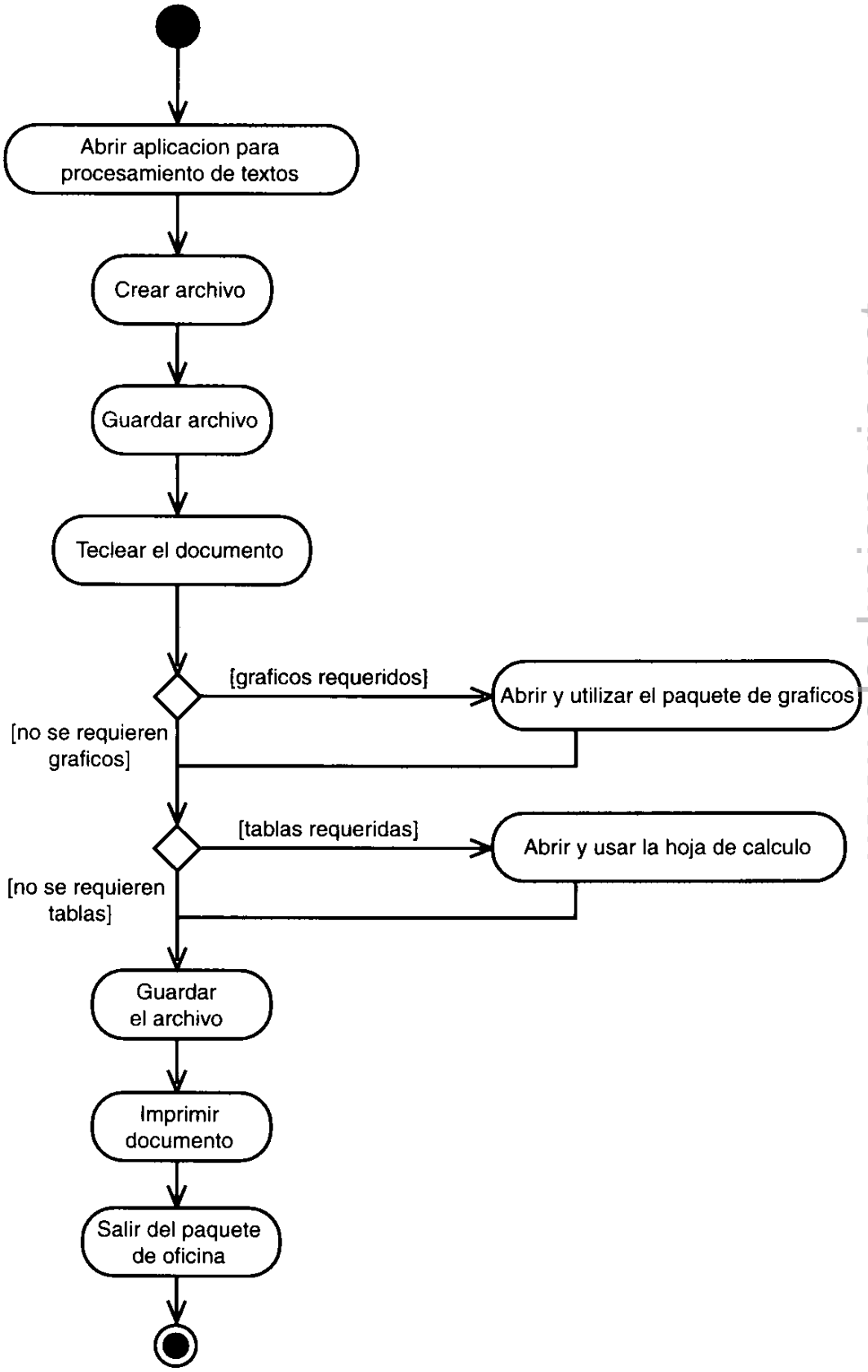
www.elsolucionario.net

7. Se guarda el archivo.
8. Se imprime el documento.
9. Se sale de la aplicación de oficina.

El diagrama de actividades de esta secuencia aparece en la figura 11.6.

**FIGURA 11.6**

*Un diagrama de actividades para el proceso de creación de un documento.*




www.elsolucionario.net

# Marcos de responsabilidad

Uno de los aspectos más útiles del diagrama de actividades es su facultad para expandirse y mostrar quién tiene las responsabilidad en un proceso.

Veamos el caso de una firma de consultoría y el proceso de negociación involucrado en una junta con un cliente. Las actividades podrían ocurrir como sigue:

1. Un vendedor hace una llamada al cliente y concierta una cita.
2. Si la cita es en la oficina del consultor, los técnicos corporativos prepararán una sala de conferencias para hacer una presentación.
3. Si es en la oficina del cliente, un consultor preparará una presentación en una laptop.
4. El consultor y el vendedor se reunirán con el cliente en el sitio y a la hora convenidos.
5. El vendedor crea una minuta.
6. Si la reunión ha planteado la solución de un problema, el consultor creará una propuesta y la enviará al cliente.

Un diagrama de actividades estándar podría lucir como en la figura 11.7.

**TERMINO NUEVO** El diagrama de actividades agrega la dimensión de visualizar responsabilidades. Para ello, separará el diagrama en segmentos paralelos conocidos como *marcos de responsabilidad.* Cada marco de responsabilidad muestra el nombre de un responsable en la parte superior, y presenta las actividades de cada uno. Las transiciones pueden llevarse a cabo de un marco a otro. La figura 11.8 muestra la versión con marcos de responsabilidad del diagrama de actividades de la figura 11.7.

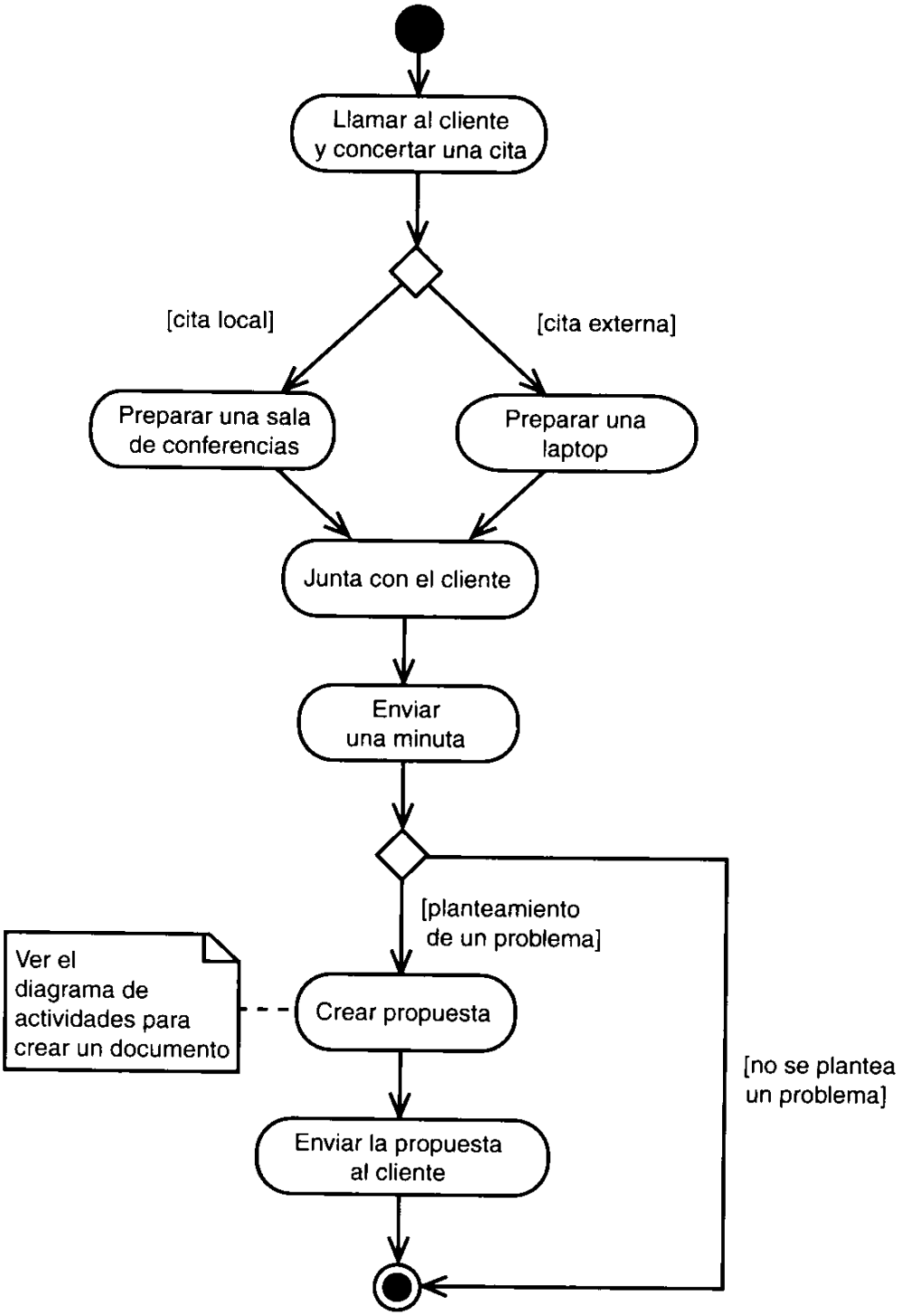
Ambos diagramas de actividades de "Reunión con un cliente nuevo" muestran la creación de una propuesta como actividad. En cada caso, tal actividad podría tener una nota adjunta que cite al diagrama de actividades para la creación del documento.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 11.7**

*Un diagrama de actividades para el proceso de negociación en una junta con un cliente.*

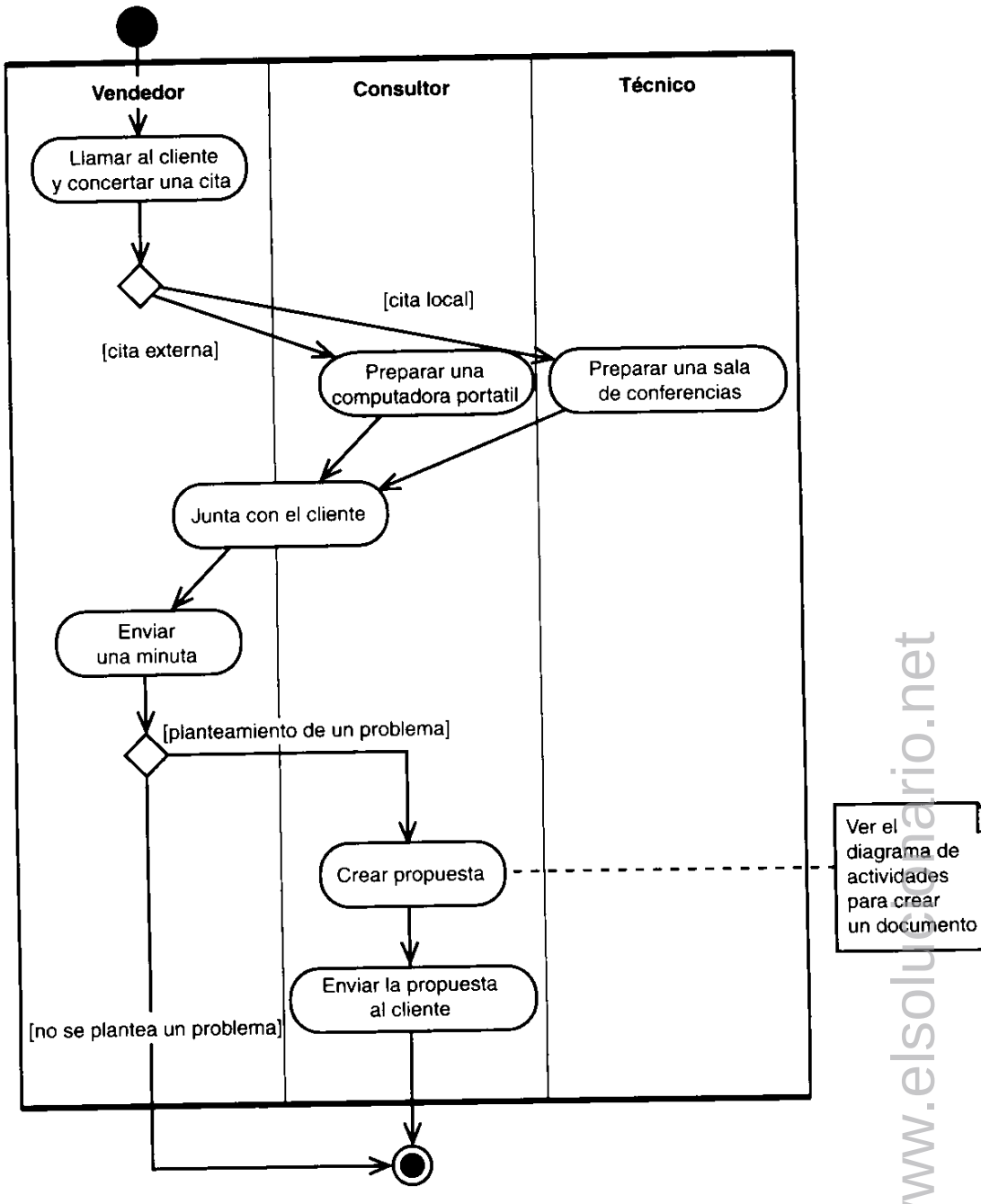



www.elsolucionario.net

www.elsolucionario.net

**FIGURA 11.8**

*La versión con marcos de trabajo de diagrama de actividades de la figura 11.7 que muestra quién es el responsable de cada actividad.*




# Diagramas híbridos

Recordemos el diagrama de actividades para la creación de un documento. Podrá depur la actividad de la impresión de un documento. En lugar de sólo mostrar una actividad "Imprimir documento", podría ser un poco más específico. La impresión se realiza dado que una señal dentro del archivo de documento se transmite desde la aplicación para el procesamiento de textos a la impresora, misma que la recibe y la imprime.

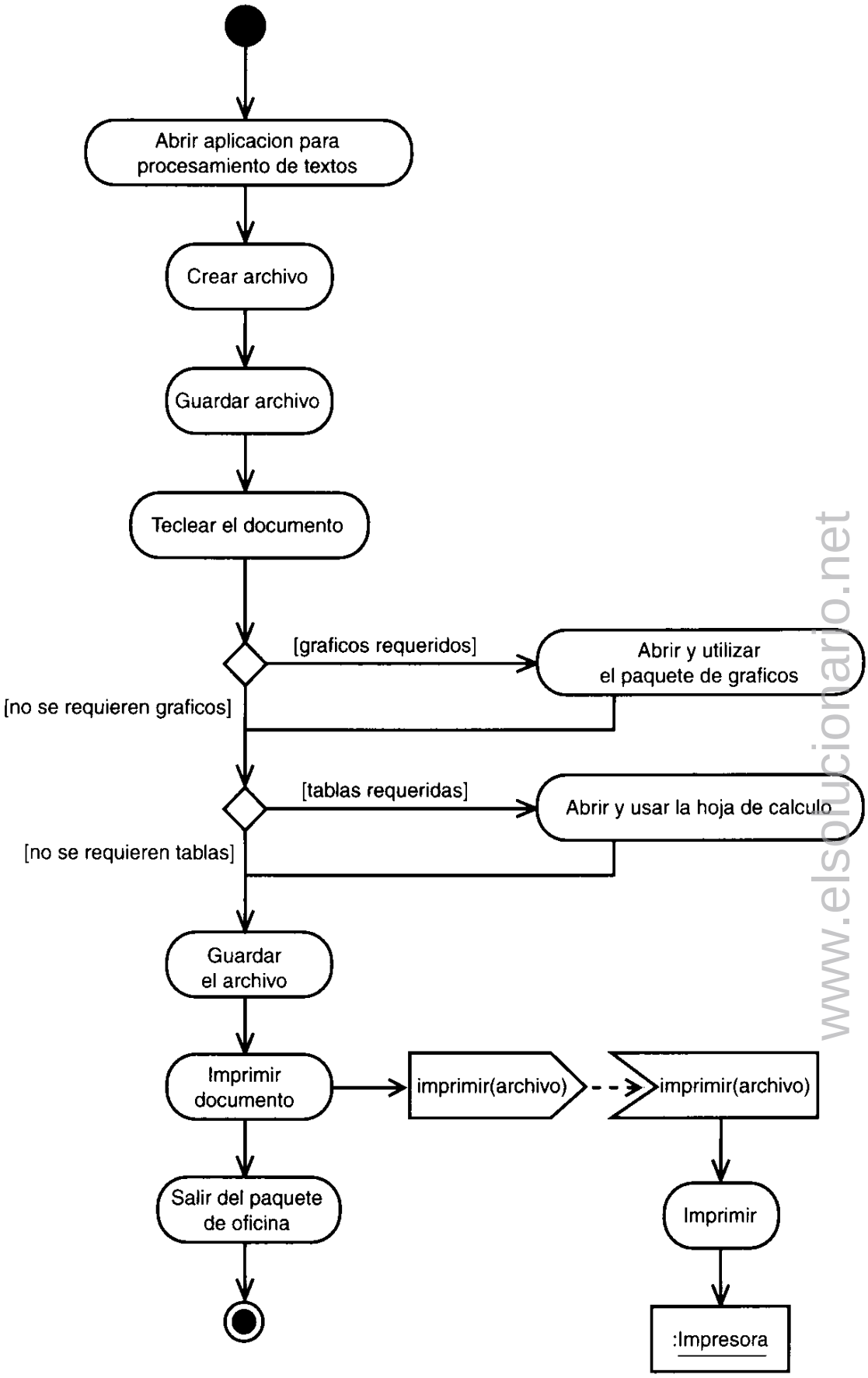
La figura 11.9 le muestra que podrá representar esto con los símbolos para la transmisio y recepción de señales, junto con un objeto Impresora que reciba al símbolo y realice su tarea de impresión. Éste es un ejemplo de diagrama híbrido, dado que contiene símbolo que normalmente asociaría con diferentes tipos de diagramas.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 11.9**

*La depuración de la actividad "Imprimir documento" nos otorga un diagrama híbrido.*

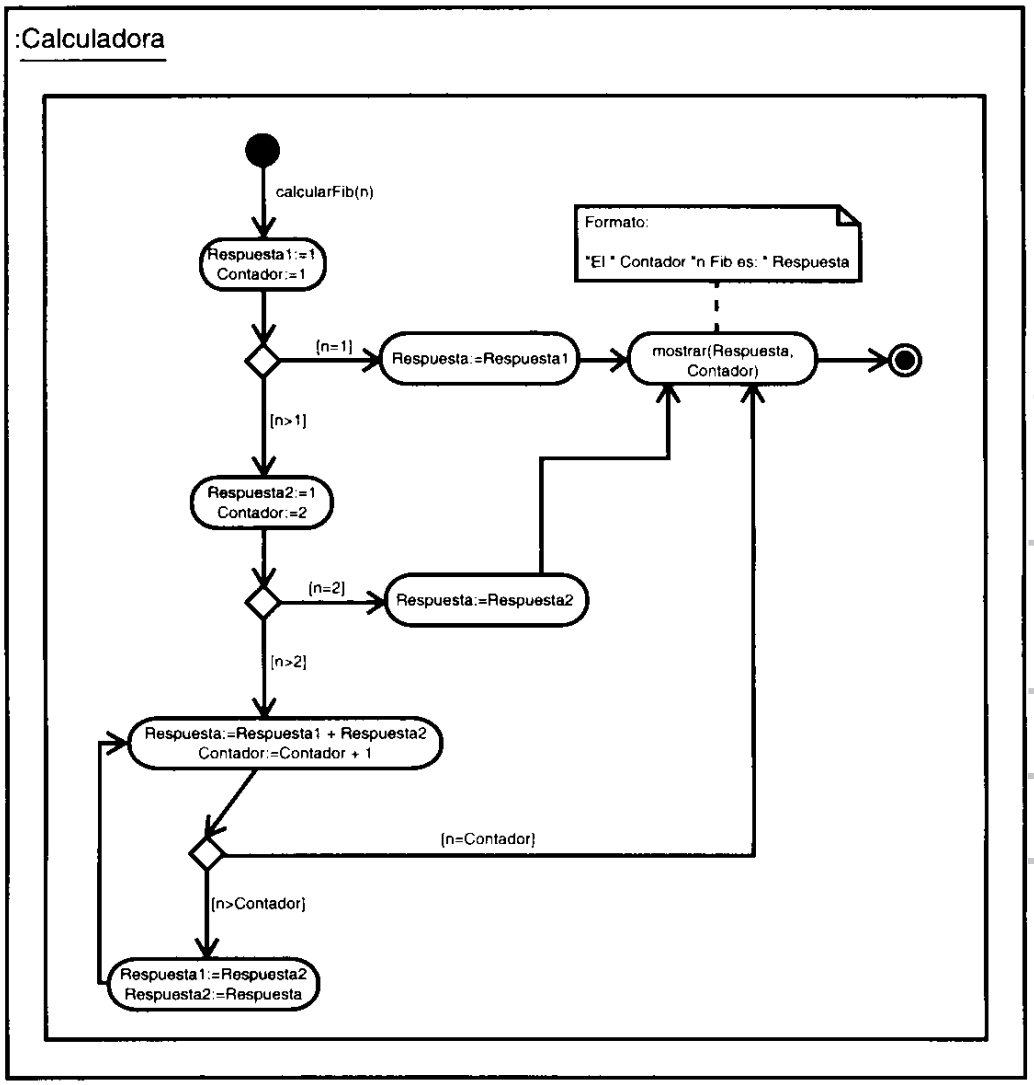



www.elsolucionario.net

He aquí otra posibilidad de diagrama híbrido: podrá mostrar un diagrama de actividades para realizar una operación dentro de un símbolo de objeto, y mostrar al objeto que recibe una petición para ejecutar la operación. Suponga que modeló al objeto Calculadora, el cual calcula los números de Fibonacci. Los desarrolladores podrían encontrar útil que usted lo representara con un diagrama híbrido como el de la figura 11.10.

**FIGURA 11.10**

*Un diagrama híbrido puede mostrar un diagrama de actividades dentro de un objeto.*




www.elsolucionario.net

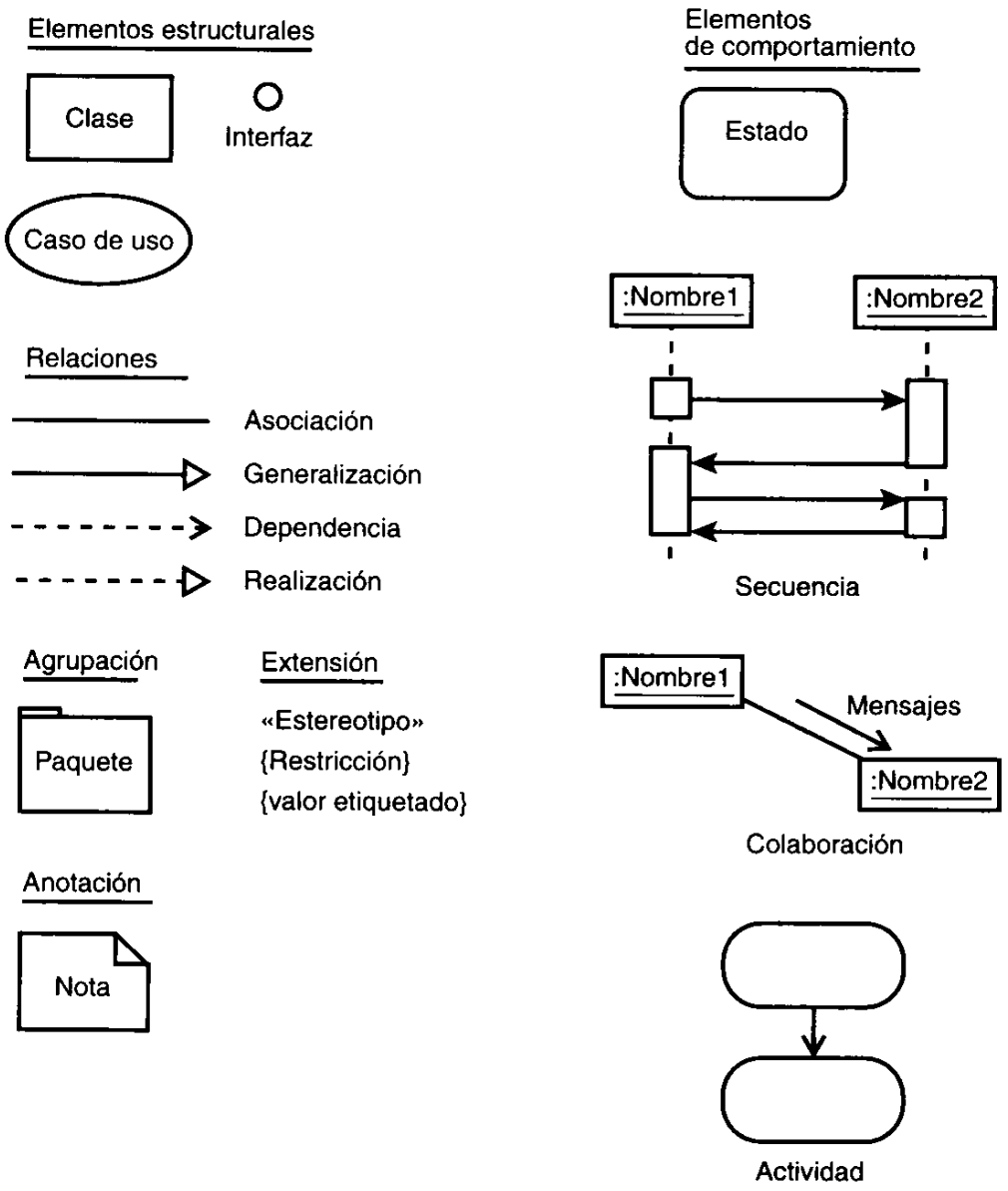
# Adiciones al panorama

La figura 11.11 muestra el panorama del UML, donde se incluye el diagrama de actividades. Este diagrama es un elemento de comportamiento.

www.elsolucionario.net

**FIGURA 11.11**

*El panorama del UML ahora incluye al diagrama de actividades.*




# Resumen

El diagrama de actividades del UML es muy parecido a un diagrama de flujo. Muestra los pasos, puntos de decisión y bifurcaciones. Este tipo de diagrama es útil para representar las operaciones de un objeto y los procesos de negocios.

El diagrama de actividades es una extensión del diagrama de estados. Los diagramas de estados destacan los estados y representan actividades como flechas entre los estados. Los de actividad se enfocan en las actividades. Cada actividad se representa como un rectángulo con esquinas redondeadas, más ovalados en apariencia que la representación de un estado. El diagrama de actividades utiliza los mismos símbolos que el de estados para los puntos de inicio y final.

www.elsolucionario.net

www.elsolucionario.net

Cuando una ruta se divide en dos o más, tal dispersión se representa con una línea gruesa perpendicular a las rutas, mismas que se reúnen en una línea similar. Dentro de un diagrama de secuencias puede mostrar una señal, cuya transmisión se representa con un pentágono convexo, y la recepción con uno cóncavo.

En un diagrama de actividades, puede representar las actividades de acuerdo con la responsabilidad asignada. Esto lo haría con marcos de responsabilidad, mismos que son segmentos paralelos que corresponden a los responsables de realizar cada tarea.

Es posible combinar al diagrama de actividades con símbolos de otros diagramas con lo que se producirán diagramas híbridos.

# Preguntas y respuestas

**P** **Ésta es otra de esas preguntas de "¿Realmente lo necesito?". Con todo lo que nos muestra un diagrama de estados, ¿realmente necesito los de actividad?**

**R** Yo le recomiendo que incluya los diagramas de actividades en su análisis. Pueden poner en claro algunos procesos y operaciones para usted y sus clientes. También son muy útiles para los desarrolladores. Es muy probable que un buen diagrama de actividades sea de gran utilidad para que un desarrollador codifique una operación.

**P** **Ha mostrado dos tipos de diagramas híbridos. ¿El UML establece limitaciones en los tipos de híbridos que puede crear?**

**R** No, en absoluto. El UML no intenta ser restrictivo. Aunque tiene algunas reglas sintácticas, la idea es que los analistas generen un modelo que transmita una idea consistente a los clientes, diseñadores y desarrolladores; no la de satisfacer ceñidas reglas lingüísticas. Si puede generar un diagrama híbrido que ayude a que todos los involucrados comprendan un sistema, hágalo.

# Taller

El cuestionario y los ejercicios le harán razonar los diagramas de actividades y su utilidad. Las respuestas se encuentran en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿Cuáles son las dos formas de representar a un punto de decisión?
2. ¿Qué es un marco de responsabilidad?
3. ¿Cómo representaría la transmisión y recepción de una indicación?

www.elsolucionario.net

www.elsolucionario.net

## Ejercicios

1. Cree un diagrama de actividades que muestre el proceso que realizaría al encender su automóvil. Empiece al colocar la llave en el switch, finalice con el motor en funcionamiento y tome en cuenta todo lo que haría si el motor no arrancara de inmediato.
2. ¿Qué podría agregar al diagrama de actividades en el proceso de negociación de la junta con un cliente nuevo?
3. Si distribuye tres piedras de modo que una de ellas se ubique en una fila y las otras dos en otra, formarían un triángulo. Si hace lo mismo con seis piedras, de manera que se ubiquen: una en una fila, dos en la siguiente, y tres en la última, también formarían un triángulo. Por ello, a los número 3 y 6 se les conoce como *números triangulares*. El siguiente número triangular es 10, el que le sigue es 15, y así por el estilo. El primer número triangular es 1. Cree dos diferentes diagramas de actividades para un proceso que calcule el enésimo número triangular. Para uno, inicie con *n* y continúe en orden regresivo. Para el otro, inicie con 1 y continúe en orden progresivo.
4. He aquí un ejercicio para quienes les gustan las matemáticas. Si se sintió a gusto con el ejercicio 3, tal vez le guste este otro. Si no, tan sólo continúe con la siguiente hora (¡podría intentar diagramar lo que dije en las dos últimas oraciones!). En la geometría coordinada, se representa un punto en el espacio al mostrar su posición x y y. Por ello, puede decir que la ubicación del punto 1 es X1, Y1. La del punto 2 es X2, Y2. Para encontrar la distancia entre estos puntos, eleve al cuadrado X2-X1 y luego Y2-Y1. Sume ambos cuadrados y calcule la raíz cuadrada de la suma. Cree un diagrama de actividades para una operación *distancia* (*X1, Y1, X2, Y2*) que localice la distancia entre dos puntos.

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 12

# Diagramas de componentes

En las horas anteriores ha visto diagramas que tratan temas conceptuales. Un diagrama de clases representa un concepto, es decir, la abstracción de elementos que confluyen en una categoría; un diagrama de estados también representa un concepto, como son los cambios en el estado de un objeto.

Ahora verá el uso de un diagrama que es algo distinto a los que ha visto, y aprenderá lo correspondiente a un diagrama UML que representa a una entidad real: un componente de software.

En esta hora se tratarán los siguientes temas:

- Qué es un componente
- Componentes e interfaces
- Qué es un diagrama de componentes
- Aplicación de los diagramas de componentes
- Diagramas de componentes en el panorama del UML

www.elsolucionario.net

www.elsolucionario.net

# Qué es un componente

Un componente de software es una parte física de un sistema, y se encuentra en la computadora, no en la mente del analista. ¿Qué puede tomarse como componente? Una tabla, archivo de datos, ejecutable, biblioteca de vínculos dinámicos, documentos y cosas por el estilo.

¿Cuál es la relación entre un componente y una clase? Imagine a un componente como la personificación en software de una clase. La clase representa una abstracción de un conjunto de atributos y operaciones. Un importante punto por recordar de los componentes y clases es que un componente puede ser la implementación de más de una clase.

Si el componente se encuentra en una computadora y es la parte funcional del sistema, ¿para qué preocuparse por él? Tendrá que modelar componentes y sus relaciones para que:

1. Los clientes puedan ver la estructura del sistema finalizado.
2. Los desarrolladores cuenten con una estructura con la cual trabajar en adelante.
3. Quienes escriban las notas técnicas y la documentación puedan entender de qué escribirán.
4. Usted se aliste para volver a utilizar los componentes.

Exploremos el último punto. Uno de los aspectos más importantes de los componentes es el potencial que tienen de volver a ser utilizados. Con las necesidades actuales de los negocios de soluciones rápidas, entre más rápido presente un sistema para producción, mayor será su competitividad. Si puede crear un componente para un sistema y puede volver a utilizarlo en otro, usted habrá contribuido a esa competitividad. Tómese el tiempo y esfuerzo para modelar un componente que lo ayude a que esta reutilización pueda llevarse a cabo.

Recordaremos la reutilización al final de la siguiente sección.

www.elsolucionario.net

# Componentes e interfaces

Cuando trate con los componentes, tendrá que tratar con sus interfaces; al tratar el tema de las clases y los objetos, hablé de las interfaces. Como recordará en la hora 2, "Orientación a objetos", un objeto oculta al mundo exterior lo que hace (lo que se conoce como *encapsulación*, *encapsulamiento* u *ocultamiento de información*). El objeto tiene que presentar un "rostro" al mundo exterior, para que los demás objetos (incluso, potencialmente, los humanos) puedan pedirle que ejecute sus operaciones. A este "rostro" se le conoce como *interfaz* del objeto.

TERMINO NUEVO

Di mayores detalles de esta idea en la hora 5, "Agregación, composición, interfaces y relación". Como lo mencioné en su momento, diversas clases podrían no estar relacionadas con una clase principal (como en la herencia), pero sus acciones podrían incluir algunas de las mismas operaciones con las mismas firmas. Podrá reutilizar

www.elsolucionario.net

este conjunto de operaciones de clase en clase. La *interfaz* es la construcción UML que le permite hacer esto. Una interfaz es un conjunto de operaciones que especifica algo respecto al comportamiento de una clase. Imagine a una interfaz como una clase que sólo contiene operaciones (no atributos). Para resumir: la interfaz es un conjunto de operaciones que presenta una clase a otras.

Al tratar las interfaces en la hora 5, también mencioné que la relación entre una clase y su interfaz se conoce como *realización*.

¡Un momento! Parecería que modelar una interfaz es la práctica de modelar un concepto. Al principio de esta hora, le dije que cuando modele un componente no será algo conceptual, dado que se encontrará en una computadora. ¿Dónde está la conexión?

En sí, una interfaz puede ser física o conceptual. La interfaz que utiliza una clase es la misma que la que utiliza su implementación de software (un componente). Como modelador, esto significa que la de la misma forma en que represente una interfaz para una clase representará una interfaz para un componente. Aunque la simbología del UML distingue entre una clase y un componente, no hace distinción entre una interfaz conceptual y una física.

TERMINO NUEVO

Hay un punto importante a este respecto: sólo podrá ejecutar las operaciones de un componente a través de su interfaz. De la misma manera que en el caso de una clase y su interfaz, la relación entre un componente y su interfaz se conoce como *realización*.

TERMINO NUEVO

Hay otro punto por destacar: un componente puede hacer disponible su interfaz para que otros componentes puedan utilizar las operaciones que contiene. Es decir, un componente puede acceder a los servicios de otro componente. El componente que proporciona los servicios se dice que provee una *interfaz de exportación*. Al que accede a los servicios se dice que utiliza una *interfaz de importación*.

## Sustitución y reutilización

Las interfaces se destacan de forma importante en los conceptos primordiales de sustitución y reutilización de componentes. Puede sustituir un componente con otro si el nuevo contiene las mismas interfaces que el anterior. Podrá reutilizar un componente en otro sistema si éste puede acceder al componente reutilizado mediante sus interfaces. Puede diseñar un componente para ser reutilizado en proyectos de desarrollo a lo largo de su empresa si quiere depurar sus interfaces para que un amplio rango de componentes puedan acceder a ellos.

Es aquí donde son útiles las interfaces en el modelado. Puede simplificarse la vida de un desarrollador que intente sustituir o reutilizar un componente si la información de su interfaz se encuentra disponible como un modelo. Si no, el desarrollador tendrá que pasar por el largo proceso de hacer un seguimiento del código.

www.elsolucionario.net

www.elsolucionario.net

# Tipos de componentes

Conforme avance en su carrera como modelador, se encontrará con tres tipos de componentes:

1. *Componentes de distribución*, que conforman el fundamento de los sistemas ejecutables (por ejemplo, DLL, ejecutables, controles ActiveX y Java Beans).
2. *Componentes para trabajar en el producto*, a partir de los cuales se han creado los componentes de distribución (como archivos de base de datos y de código).
3. *Componentes de ejecución*, creados como resultado de un sistema en ejecución.

Si es usted un usuario de Windows, encontrará ejemplos de los tres tipos de componentes cuando utilice la ayuda, aunque tal vez no lo sepa. El componente de distribución es el archivo .HLP (Archivo de Ayuda): Cuando haga clic en uno, se abrirá el cuadro de diálogo de los temas de la ayuda y empezará a buscar el tema que elija. La primera vez que haga clic en la ficha Buscar, verá una pequeña animación que le indicará que se está creando un índice. Un archivo .CNT (tema de contenido) describe el esquema del contenido. Dado que este archivo le ayuda a crear un componente de distribución (puede utilizar la página de índice por siempre), es un componente para trabajar en el producto. A su vez, el índice creado se encuentra en un archivo .FTS (búsqueda de texto completo). Finalmente, la primera vez que abra la ayuda, se creará un archivo .GID (índice general), que es resultado de un análisis del sistema de ayuda de Windows que agiliza el acceso a los temas del archivo de ayuda. Con ello, los archivos .FTS y .GID son componentes de ejecución.

# Qué es un diagrama de componentes

Un diagrama de componentes contiene, obviamente, componentes, interfaces y relaciones. También pueden aparecer otros tipos de símbolos que ya haya visto.

## Representación de un componente

El símbolo principal de un diagrama de componentes es un rectángulo que tiene otros dos sobrepuestos en su lado izquierdo. La figura 12.1 le muestra este símbolo. Debe colocar el nombre del componente dentro del símbolo. El nombre es una cadena.

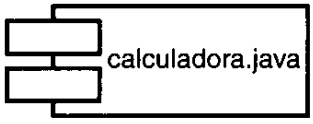


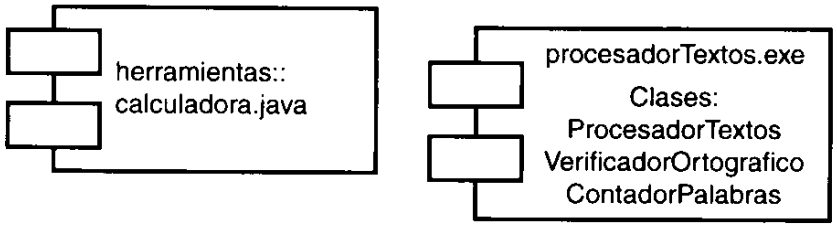

**FIGURA 12.1**
*El símbolo que representa a un componente.*

www.elsolucionario.net

La figura 12.2 le muestra que si el componente es miembro de un paquete, puede utilizar el nombre del paquete como prefijo para el nombre del componente. También puede agregar información que muestre algún detalle del componente.

**FIGURA 12.2**

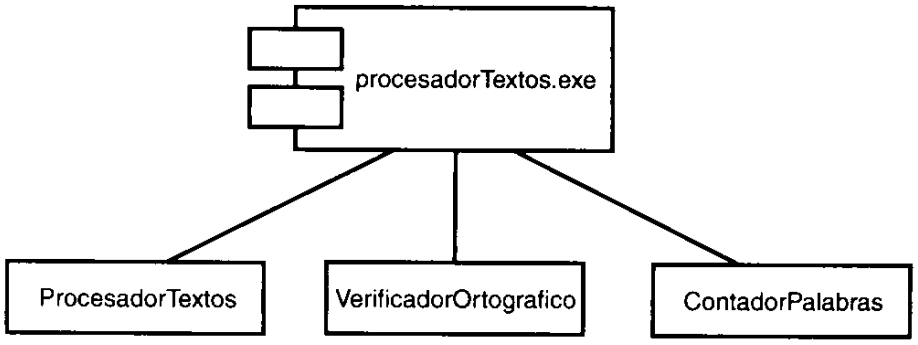
*Adición de información al símbolo del componente.*




El símbolo a la derecha de la figura 12.2 le muestra las clases que implementa un componente en particular. La figura 12.3 le muestra otra forma de hacer esto, aunque esta técnica por lo general desordena el diagrama. Vea las relaciones de dependencia entre el componente y las clases.

**FIGURA 12.3**

*Símbolos de las relaciones entre un componente y las clases que implementa.*

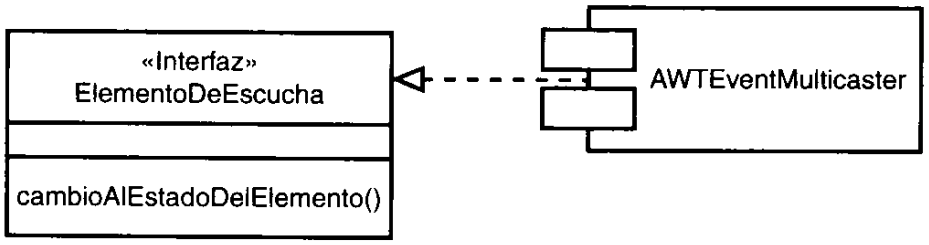



## Cómo representar las interfaces

Existen dos formas para representar a un componente y sus interfaces: la primera muestra la interfaz como un rectángulo que contiene la información que se le relaciona, se conecta al componente por la línea discontinua y una punta de flecha representada por un triángulo sin rellenar que visualiza la realización (vea la figura 12.4).

**FIGURA 12.4**

*Puede representar a una interfaz como un rectángulo con información, conectado al componente por una flecha de realización.*




www.elsolucionario.net

www.elsolucionario.net

La figura 12.5 le muestra la segunda forma de representar a un componente y sus interfaces; esta forma es representativa, ya que representará la interfaz como un pequeño círculo que se conecta al componente por una línea continua. En este contexto, la línea representa la relación de realización (compare a las figuras 12.4 y 12.5 con las figuras 5.6 y 5.7).

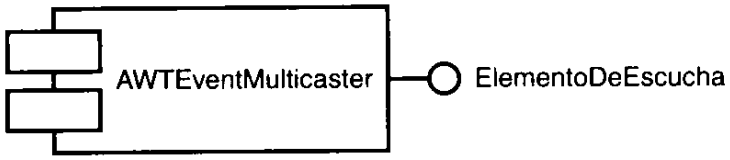



**FIGURA 12.5**

*Puede representar a una interfaz como un pequeño círculo, conectado al componente por una línea continua que, en este contexto, se interpreta como realización.*

Además de la realización, puede representar a la dependencia, que es la relación entre un componente y una interfaz de importación. Como recordará, la dependencia se vislumbra como una línea discontinua con una punta de flecha. Puede mostrar la realización y la dependencia en el mismo diagrama, como se ve en la figura 12.6.

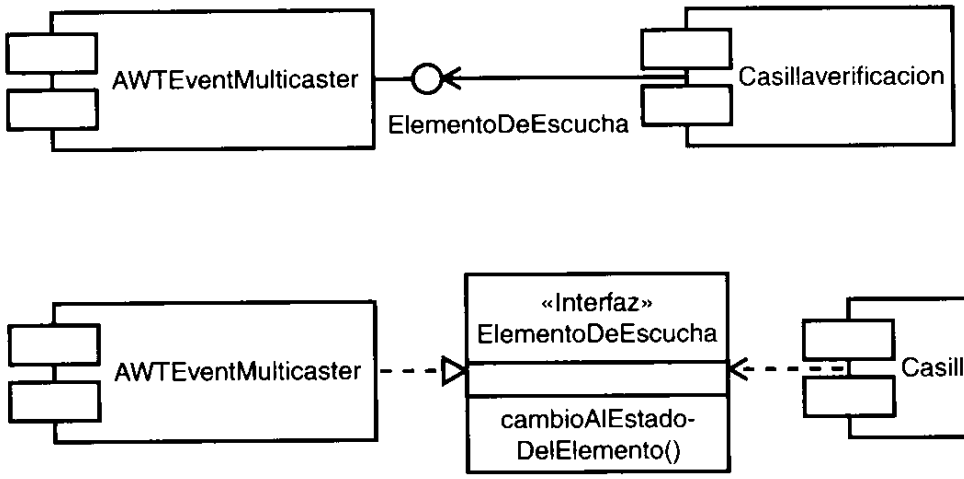



**FIGURA 12.6**

*Una interfaz que realiza un componente y otra de la que depende.*

# Aplicación de los diagramas de componentes

Algunos ejemplos le ayudarán a empezar a usar los diagramas de componentes. El primero es un modelo de una página Web que representa a un componente Java. El siguiente también es un modelo de una página Web pero éste utiliza controles ActiveX. Finalizará con un modelo de un paquete de Microsoft conocido como PowerToys. Este paquete, que puede obtener del sitio de Microsoft, le permite modificar aspectos de Win32.

## Una página Web con un subprograma Java

Este ejemplo modela un programa tomado del excelente y entretenido libro de Rogers Cadenhead llamado *Aprendiendo programación con Java 1.1 en 24 horas* (Prentice Hall

www.elsolucionario.net

Hispanoamericana, 1997). El ejemplo aparece en la hora 22, "Escriba un juego para Web". Rogers muestra cómo generar un applet (o subprograma) que ejecuta el juego de dados "Craps" en una página Web, y utiliza una clase llamada "Die" (para crear los dados) de la hora 21, "A jugar con Java". Para ver los detalles del código, tendrá que leer el libro. Aquí sólo nos concentraremos en los componentes.

**TÉRMINO NUEVO** Un applet es un pequeño programa hecho en Java que funciona en una página Web.

La página Web se llama Craps.html. El código fuente del applet se encuentra en el archivo Craps.java, y el código objeto es el archivo Craps.class. El código fuente de la clase Die se encuentra en Die.java y el código objeto en Die.class. Los cinco archivos se encuentran en el mismo directorio —que llamaremos Tirodedados (que no es el nombre que Roger le dio)—.

Craps.html depende, obviamente, de Craps.class y Die.class. Cada archivo .class es un componente y cada uno es la implementación de una clase. Lo que no es muy obvio (tendría que ver el código fuente para descubrirlo) es que tanto Craps.java como Die.java importan (utilizan las clases de) java.awt, un grupo de clases que muestran y controlan la GUI (el "awt" significa: "Conjunto de herramientas abstractas para manejar ventanas").

En el contexto de Java, la *importación* permite al desarrollador utilizar sólo el nombre de una operación cuando la escribe en un programa, en lugar de utilizar toda la ruta de la operación (que podría ser muy larga). La importación no "incrusta" una clase en otra. Tan sólo permite escribir algo más corto.

Craps.java es un applet, y por ello se hereda desde la clase java.applet.Applet. Finalmente, Craps.java importa a java.awt.event e implementa una interfaz ActionListener (para responder a los eventos generados por el usuario, como hacer clic con el ratón).

Dentro del código, la interfaz ActionListener proporciona un botón para que el usuario haga clic para que se "tiren los dados". Al ver la referencia de Java, notará que la clase java.awt.AWTEventMulticaster implementa esta interfaz.

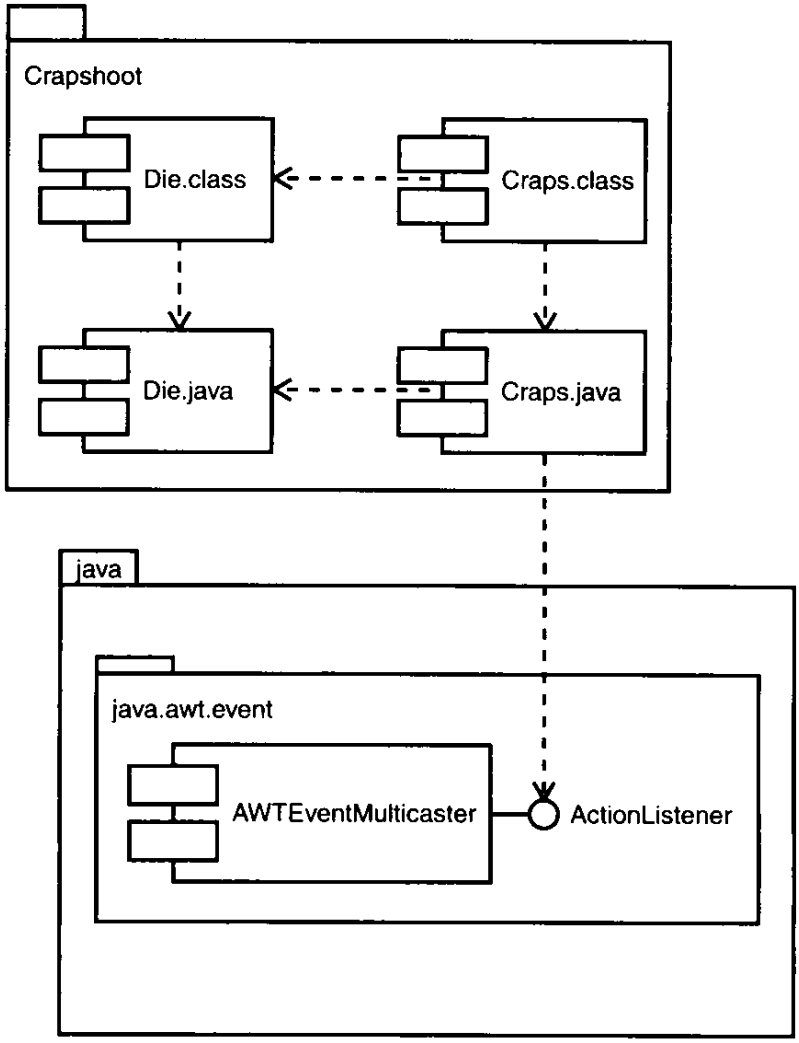
¿Sabe qué? ¡Esto sería más fácil de comprender si viera el modelo UML! La figura 12.7 le muestra el diagrama de componentes. Un paquete corresponde al directorio en donde se encuentran los archivos, el otro al JDK (Conjunto de herramientas para desarrollo en Java).

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 12.7**

*El diagrama para el juego de dados basado en la Web de Rogers Cadenhead.*



**TERMINO NUEVO** Este ejercicio —generar un modelo a partir de un código existente— se conoce como *ingeniería inversa.*

## Una página Web con controles ActiveX

ActiveX es el medio de Microsoft para agregar componentes a las aplicaciones. Con tantos tipos de componentes ActiveX (controles) disponibles, podrá encontrar alguno que haga casi todo lo que requiera una aplicación. Una propiedad de un componente ActiveX es su número de identificación hexadecimal único de 32 bits, conocido como CLSID (identificador de la clase).

Si sus requerimientos son especiales, podrá generar su propio componente ActiveX en Visual Basic o Visual C++. Luego podrá reutilizarlo de aplicación en aplicación.

En las páginas Web, los componentes ActiveX se encuentran y trabajan con el código escrito en algún lenguaje para secuencias de comandos como VBScript. En este ejemplo, la página Web cuenta con un control Timer ActiveX, dos cuadros combinados ActiveX y tres botones ActiveX. La página Web permite a un usuario establecer los parámetros para

www.elsolucionario.net

www.elsolucionario.net

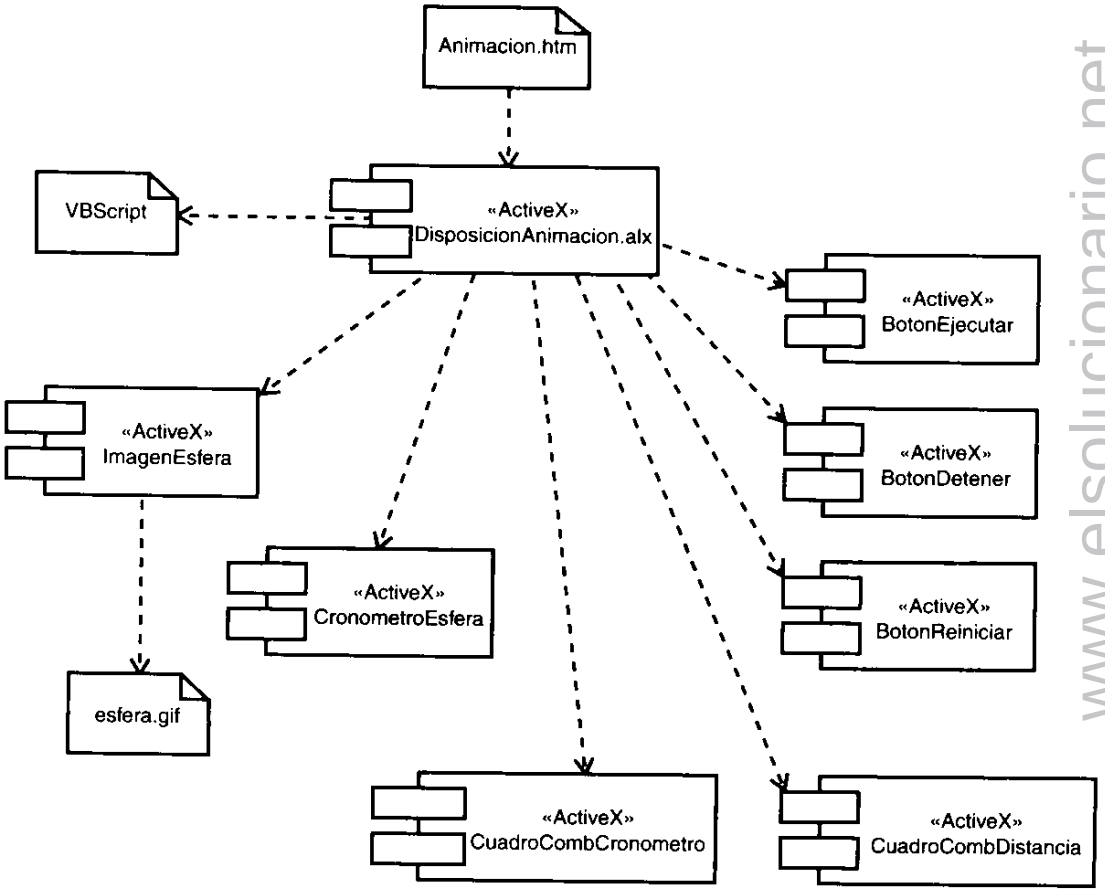
animar el movimiento de una esfera (una imagen .gif) por la pantalla. De un cuadro combinado, el usuario selecciona la cantidad de píxeles por movimiento. Del otro se selecciona la cantidad de milisegundos entre movimientos. Un botón iniciará el movimiento, el otro lo detendrá y el tercero restaurará la esfera a su posición inicial. El cronómetro moverá la esfera cuando pase la cantidad de milisegundos elegida por el usuario.

Los controles ActiveX se encuentran en un componente separado conocido como *Disposición* (Layout). La página HTML y la disposición se encuentran en el mismo directorio.

La figura 12.8 le muestra el diagrama de componentes para esta página. Observe el uso del símbolo de anotación para representar al VBScript. Aunque esto no es absolutamente necesario, destaca una diferencia entre el lenguaje de secuencias de comandos y los componentes compilados ActiveX.

**FIGURA 12.8**

*El diagrama de componentes para una página Web con componentes ActiveX.*




www.elsolucionario.net

## PowerToys

Si utiliza cualquier versión de Win32, ya conocerá las horribles flechas pequeñas que se encuentran en la esquina inferior izquierda de cada icono de acceso directo. Microsoft tiene un paquete llamado PowerToys que le permite eliminarlas y hacer varias otras cosas con la GUI, mediante una aplicación llamada TweakUI que es parte del paquete.

Puede obtener a PowerToys del sitio Web de Microsoft. Es gratis. Cuando lo obtenga y descomprima, verá varios archivos con extensiones .dll. También verá un archivo de

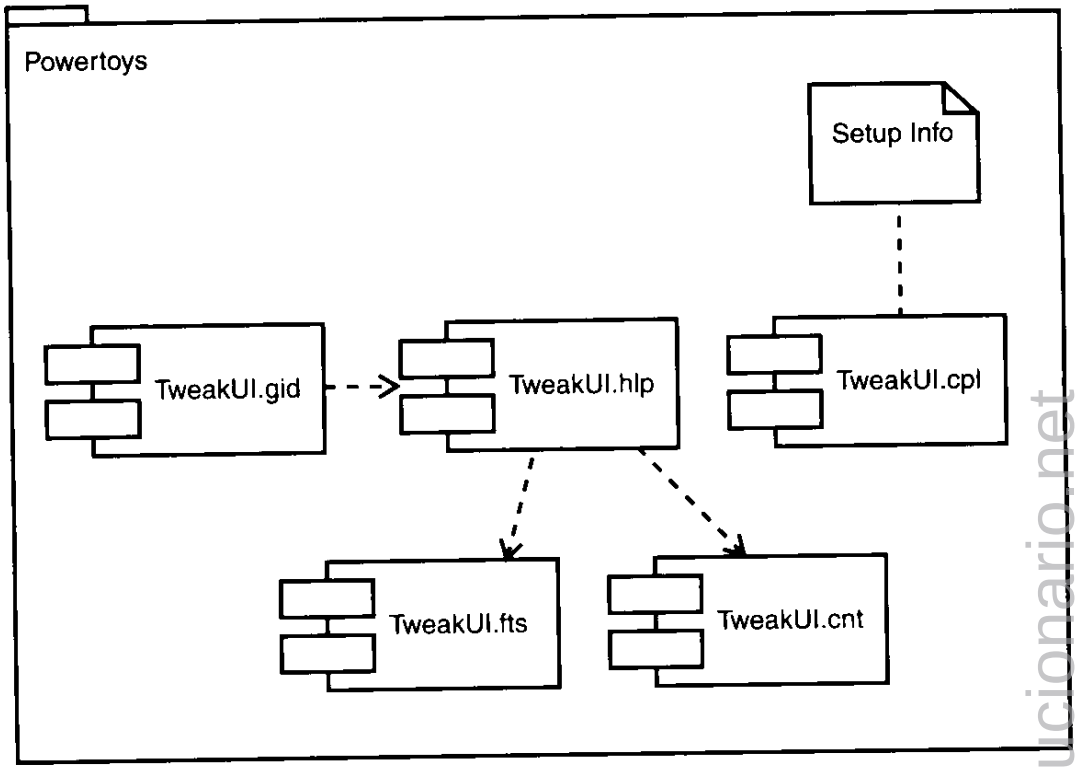
www.elsolucionario.net

ayuda y otro .CNT. Haga clic en el archivo de ayuda y generará un archivo .GID. Utilice la característica Buscar y creará un archivo .FTS.

La figura 12.9 le muestra un diagrama de componentes que modela a TweakUI en el paquete PowerToys, mismo que muestra las dependencias entre los diversos tipos de componentes.

**FIGURA 12.9**

*Modelado de TweakUI en el paquete PowerToys.*




# Diagramas de componentes en el panorama

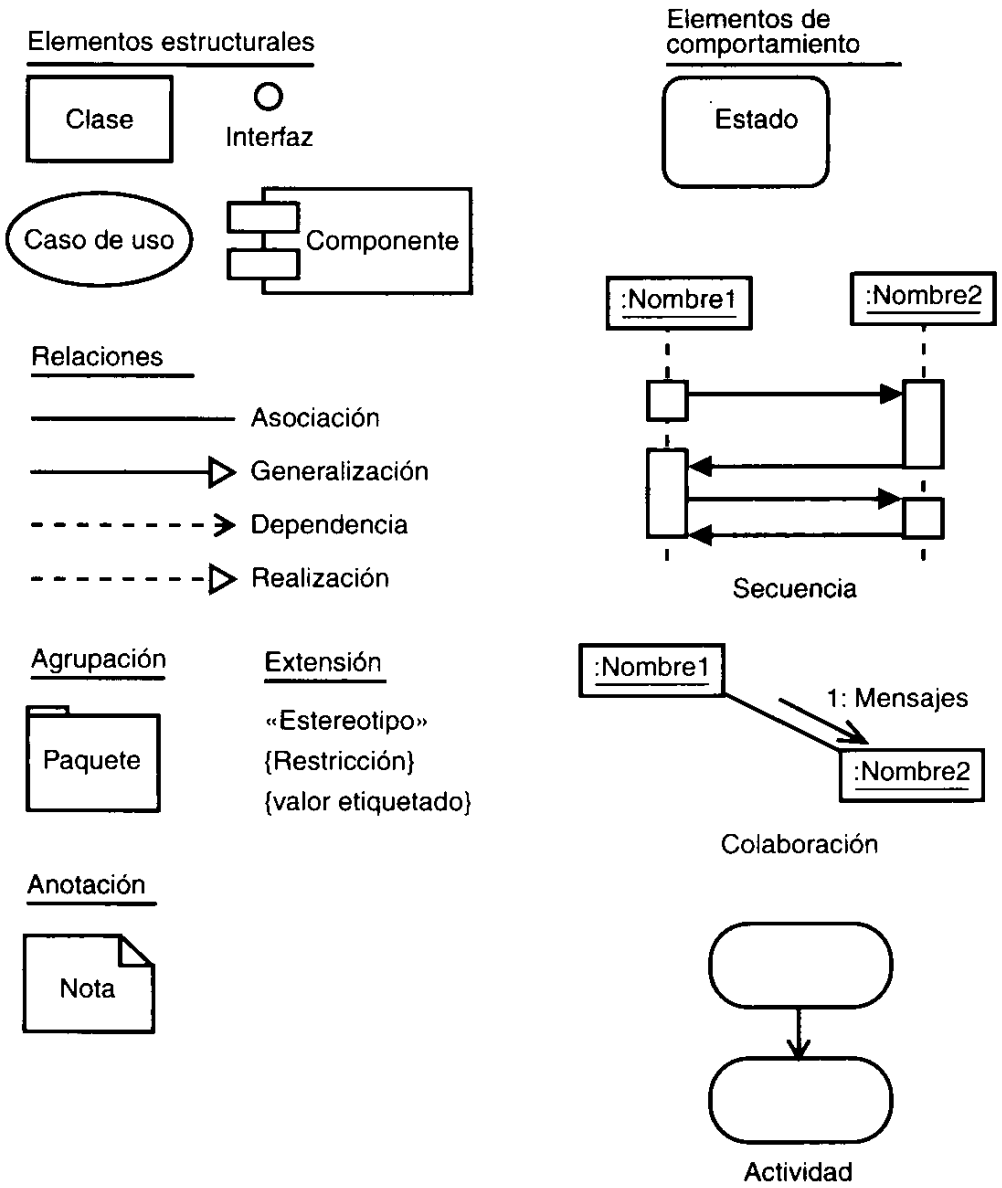
Ya casi tiene todo su panorama. La figura 12.10 incluye el diagrama de componentes, que se enfoca en una arquitectura de software del sistema. En la siguiente hora, verá cómo modelar la arquitectura de hardware.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 12.10**

*Su panorama del UML ahora incluye al diagrama de componentes.*




# Resumen

El diagrama de componentes UML es un conglomerado de figuras de los diagramas que ya ha visto. En lugar de representar una entidad conceptual como una clase o estado, un diagrama de componentes representa a un elemento real: un componente de software. Estos componentes se encuentran en las computadoras, no en la mente del analista.

Un componente puede accederse a través de su interfaz, una colección de operaciones. La relación entre un componente y su interfaz se llama *realización*. Un componente puede acceder los servicios de otro. Cuando se hace, utiliza una *interfaz de importación* . El componente que realiza la interfaz con tales servicios proporciona una *interfaz de exportación.*

La representación de un componente es un rectángulo con otros dos rectángulos pequeños sobrepuestos en su lado izquierdo. Puede representar una interfaz de dos formas: la primera es un rectángulo que contiene información de la interfaz y se conecta con el componente mediante una línea discontinua con una punta de flecha representada por triángulo sin relleno. La otra es un pequeño círculo conectado al componente con una línea continua. Ambos tipos de conexión pretenden mostrar una relación de realización.

www.elsolucionario.net

www.elsolucionario.net

# Preguntas y respuestas

**P En un diagrama de componentes, ¿cuál será la regla de oro para usar símbolos que no representen a componentes?**

**R** Esto lo hará cuando desee indicar algo que sea ciertamente distinto de un componente compilado. No es necesario, pero podría ayudar a tener otro punto de vista. Podría utilizar el símbolo de la anotación para representar archivos de encabezado, dll, o archivos de secuencias de comandos. Otra posibilidad es la de utilizar el símbolo regular del componente con un estereotipo que indique el tipo de archivo.

**P Ha utilizado a VBScript como un componente de la página Web. El código de VBScript consta de varios procedimientos. ¿No podría modelar cada uno como componente?**

**R** Sí, podría. No obstante, podría desordenar su modelo si depurara al VBScript (o JavaScript) hasta tal nivel, así que podría, mejor, agregar una nota que abunde al respecto.

# Taller

En este taller reforzará su conocimiento respecto a los componentes y cómo modelarlos. Uno de los ejercicios trata de los conceptos de una página Web, el otro de DLL. El Apéndice A, "Respuestas a los cuestionarios", será el componente del libro que contenga las respuestas.

## Cuestionario

1. ¿Cuáles son los tres tipos de componentes?
2. ¿Cómo llamaría a la relación entre un componente y su interfaz?
3. ¿Cuáles son las dos formas de representar a esta relación?
4. ¿Qué es una interfaz de exportación? ¿Que es interfaz de importación?

## Ejercicios

1. Adentrémonos en la ingeniería inversa para modelar una página Web. Visite `http://www.pearson.com.mx`, la página Web de Pearson Educación (la empresa que amablemente publicó el libro que ahora lee). Utilice el menú Ver de su explorador para seleccionar la opción que le deje a la vista el código fuente. No encontrará ningún componente ActiveX o applet de Java, pero verá diversos archivos .gif y cierto código en JavaScript. No incluya todos los archivos .gif en su modelo, sólo algunos para reafirmar su comprensión. Vaya al principio del código y verá una referencia a una hoja de estilo. La referencia se encuentra en un elemento LINK de HTML. Este es un archivo por separado que contiene la información de estilo para la página Web. Asegúrese de incluirla en su modelo.

www.elsolucionario.net

www.elsolucionario.net

2. En el diagrama de colaboraciones del caso de uso "Crear propuesta", el consultor busca en el área central de almacenamiento una propuesta adecuada para volverla a utilizar. Imagine a "buscar" como un mensaje enviado para buscar en una secuencia de archivos, y utilice las técnicas de modelado de la sección "Algunos conceptos más" para cambiar el diagrama de colaboraciones en la figura 10.6.

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 13

# Diagramas de distribución

Hasta ahora nos hemos concentrado en el entorno conceptual, aunque en la hora anterior vimos los modelos de la arquitectura de software. Es momento de concentrarnos en el hardware. Como podrá ver, hemos trascendido desde los elementos (como las clases) que se encuentran en los análisis, hasta los componentes en los equipos de cómputo y al hardware existente.

Claro está que el hardware es un tema primordial en un sistema de varios componentes. En el mundo actual de la computación, un sistema podría abarcar diversos tipos de plataformas en ubicaciones dispersas. Un diseño sólido de distribución de hardware es básico para el diseño del sistema. El UML le da los símbolos para crear una imagen clara de la forma en que deberá lucir el hardware final.

En esta hora se tratarán los siguientes temas:

- Qué es un diagrama de distribución
- Aplicación de los diagramas de distribución
- Los diagramas de distribución en el panorama del UML

www.elsolucionario.net

www.elsolucionario.net

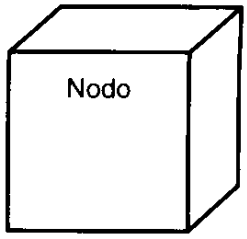
# Qué es un diagrama de distribución

**TERMINO NUEVO** El elemento primordial del hardware es un *nodo*, que es un nombre genérico para todo tipo de recurso de cómputo. Es posible usar dos tipos de nodos: un procesador, el cual puede ejecutar un componente, y un dispositivo que no lo ejecuta. Normalmente, un dispositivo (como impresora o monitor) tiene contacto de alguna forma con el mundo exterior.

En el UML, un cubo representa a un nodo. Deberá asignar un nombre para el nodo, y podrá utilizar un estereotipo para indicar el tipo de recurso que sea. La figura 13.1 le muestra un nodo.

**FIGURA 13.1**

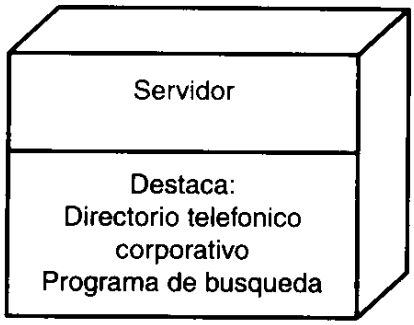
*Representación de un nodo en el UML.*




El nombre es una cadena de texto. Si el nodo es parte de un paquete, su nombre puede contener también el del paquete. Puede dividir al cubo en compartimientos que agreguen información (como componentes colocados en el nodo), como en la figura 13.2.

**FIGURA 13.2**

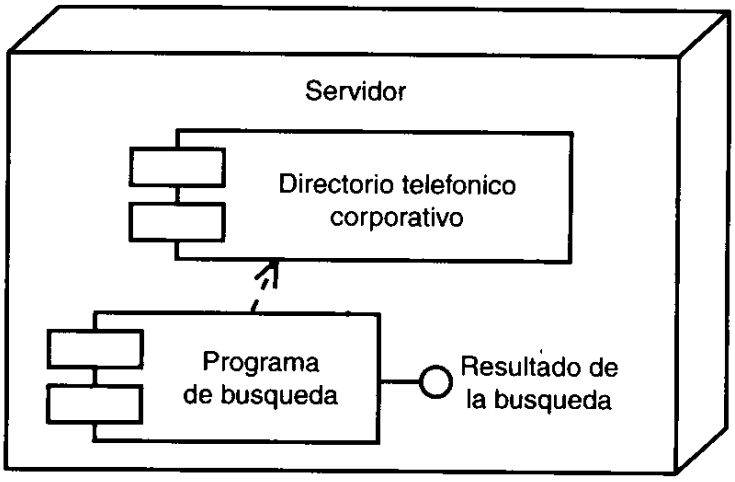
*Cómo agregar información a un nodo.*




Otra forma de indicar los componentes distribuidos es la de mostrarlos en relaciones de dependencia con un nodo (vea la figura 13.3).

**FIGURA 13.3**

*Puede mostrar los componentes en relaciones de dependencia con un nodo.*




www.elsolucionario.net

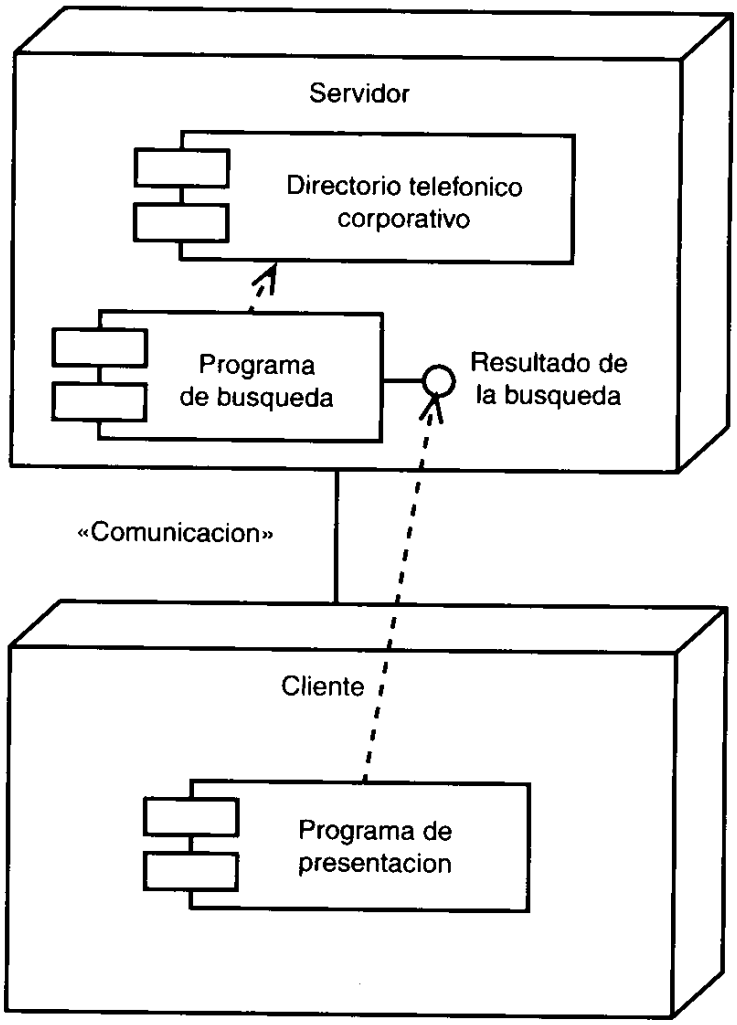
www.elsolucionario.net

Una línea que asocie a dos cubos representará una conexión entre ellos. Podrá utilizar un estereotipo para dar información respecto a la conexión. La figura 13.4 proporciona ejemplos de conexiones entre nodos.

Tenga en cuenta que una conexión no es necesariamente un cable o alambre. También puede visualizar conexiones inalámbricas como las infrarrojas o satelitales.

**FIGURA 13.4**

*Representación de conexiones entre nodos.*




Aunque la conexión es el tipo común de asociación entre dos nodos, es posible utilizar otros (como la agregación o la dependencia). Podrá representarlas de las formas ya conocidas.

# Aplicación de los diagramas de distribución

Un buen lugar para empezar es con un equipo de cómputo doméstico, por lo que el primer ejemplo es un diagrama de distribución del sistema que utilicé para escribir este libro.

www.elsolucionario.net

www.elsolucionario.net

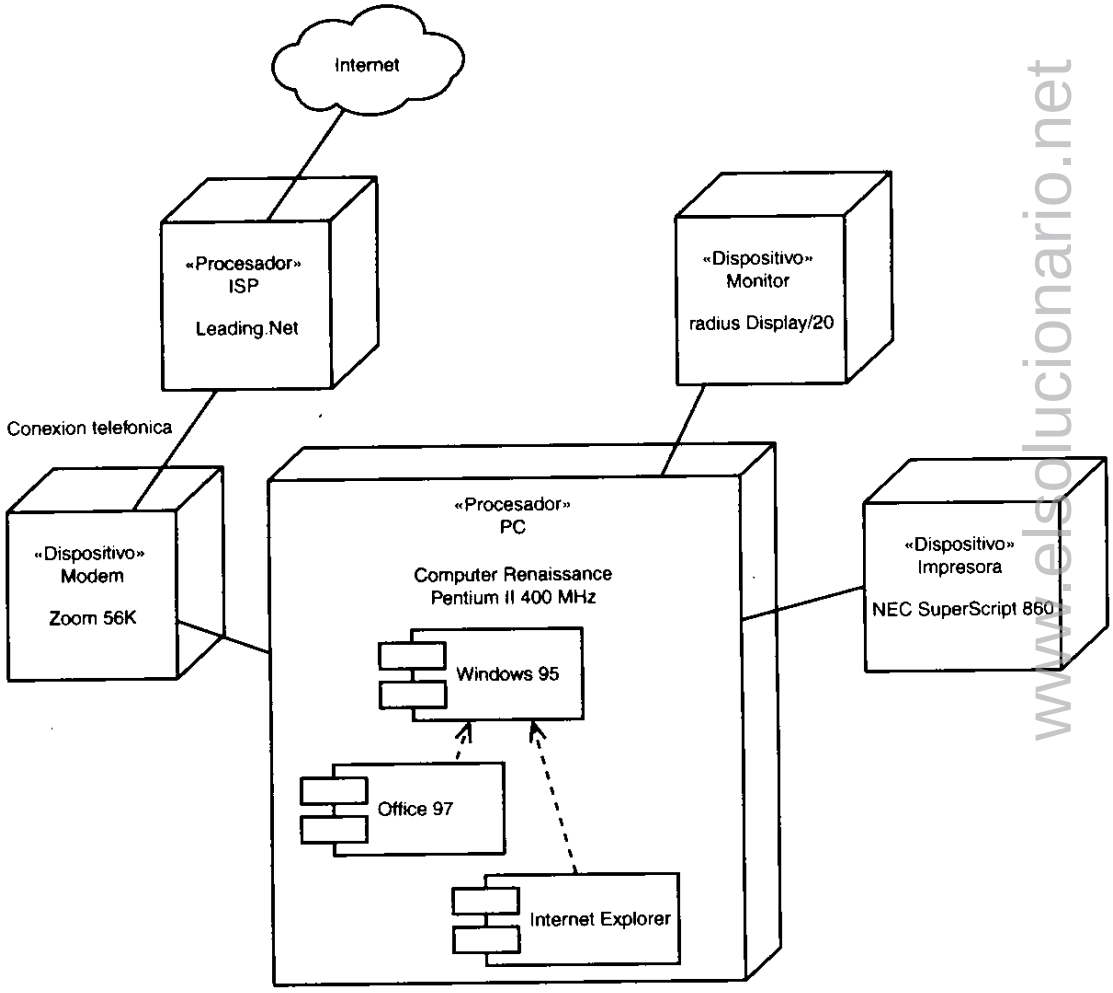
No obstante, como lo dije, los sistemas actuales de varios procesadores conectan nodos que podrían encontrarse lejanos entre sí. Para visualizar completamente este problema, necesitará también ver los ejemplos de los diagramas de distribución aplicados a las redes. Incluiré ejemplos que podrán servirle para adaptarlos a su propio entorno. Cada ejemplo incluye restricciones que reflejan las reglas de la red particular.

## Un equipo doméstico

Para modelar mi equipo de cómputo, he incluido al procesador y los dispositivos, a la vez de que he modelado mi conexión telefónica con mi proveedor de servicios de Internet y su conexión. La nube que representa la Internet no es parte de la simbología del UML, pero es útil para clarificar el modelo. La figura 13.5 presenta el diagrama de distribución.

**FIGURA 13.5**

*El diagrama de distribución de mi equipo de cómputo.*




## Una red token-ring

En una red token-ring, las computadoras equipadas con una NIC (tarjeta de interfaz de red) se conectan a una MSAU (unidad central de acceso a multiestaciones). Se conectan varias MSAU en una serie que podría parecer un anillo (por ello la parte "ring" del nombre). El anillo de MSAU se combina para fungir como un policía de tránsito, mediante una señal

www.elsolucionario.net

www.elsolucionario.net

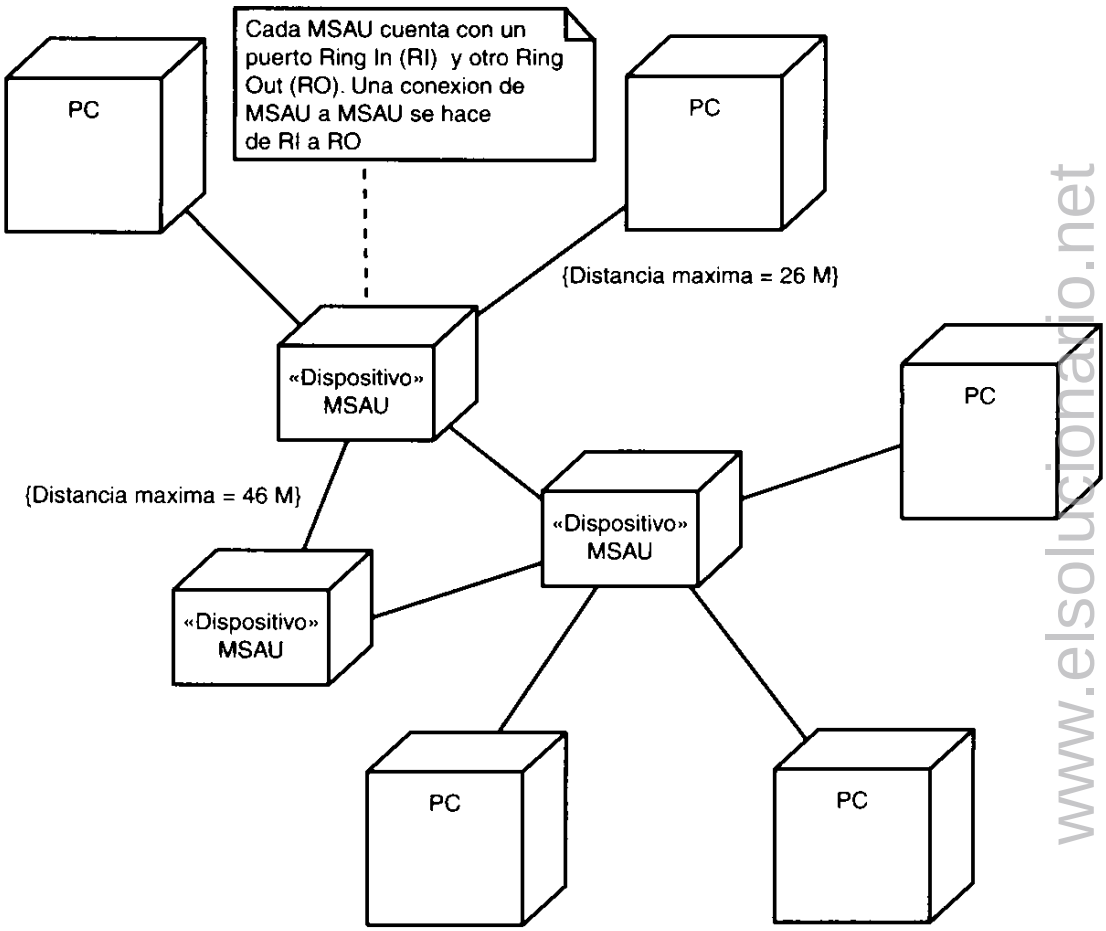
conocida como *token* que permite a cada equipo de cómputo saber cuándo puede transmitir información. Así es, el token va de equipo en equipo hasta que uno de ellos contenga información por enviar. En realidad, el token se mueve por el anillo de MSAU.

Cuando se obtiene el token, sólo esa información del equipo puede ir por la red. Una vez que se envía, la información viaja hasta su destino. Cuando llega, se devuelve un acuse de recibo al equipo que la envió.

En este ejemplo, que se aprecia en la figura 13.6, he modelado una red que consta de tres MSAU y sus respectivos equipos.

**FIGURA 13.6**

*El diagrama de distribución de una red token-ring que consta de tres MSAU.*




## ARCnet

Como en una red token-ring, una red ARCnet (Red de Cómputo de Recursos Adjuntos) implica pasar un token o señal de un equipo a otro. La diferencia es que en ARCnet cada equipo tiene asignado un número. El orden numérico determina cuál equipo obtendrá al token. Cada equipo se conecta a un concentrador o hub que podrá ser activo (amplificará la información que llega antes de transmitirla) o pasivo (transmitirá la información sin amplificarla).

www.elsolucionario.net

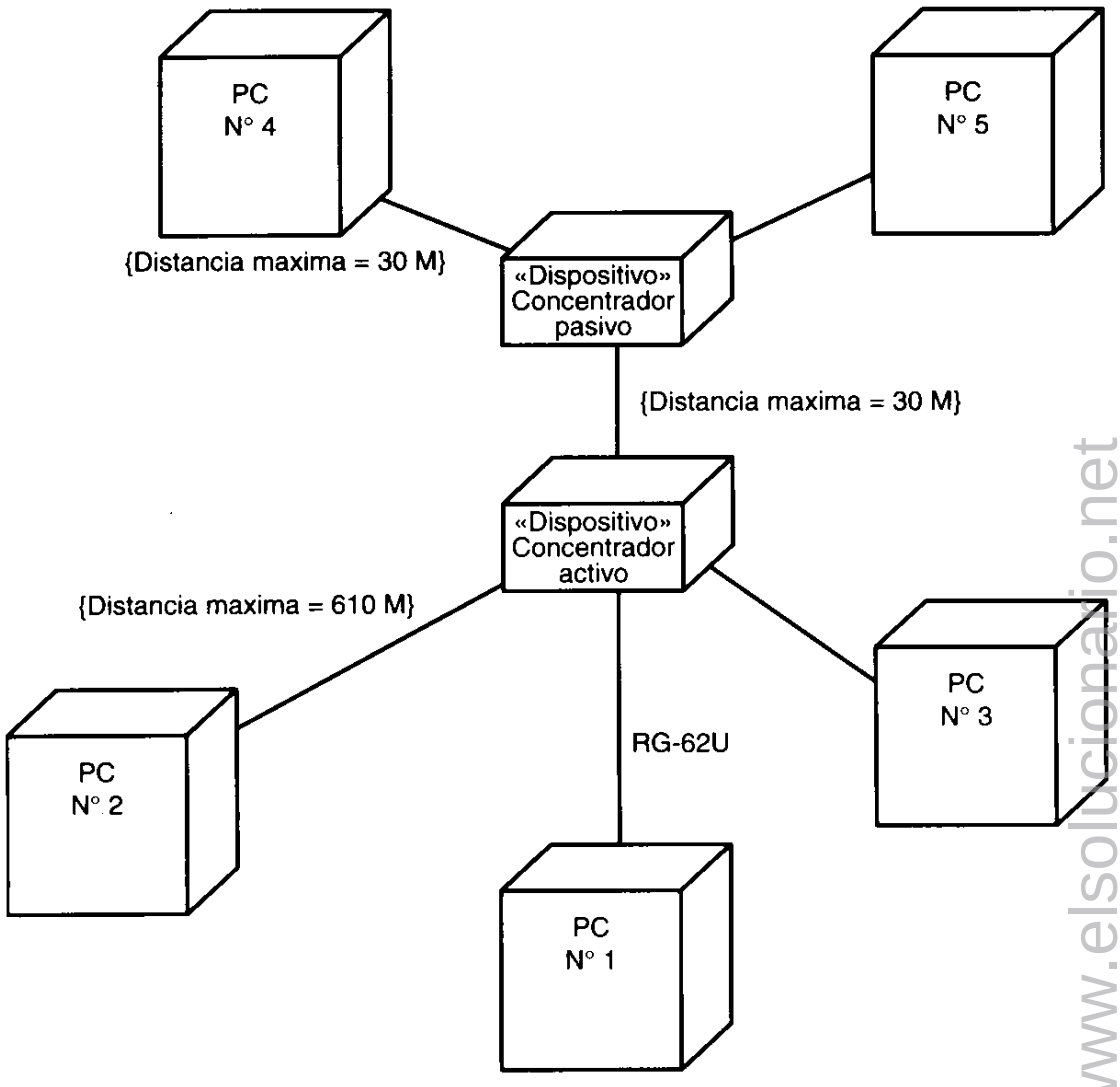
www.elsolucionario.net

A diferencia de los MSAU en una red token-ring, los concentradores ARCnet no mueven el token en un anillo. Los equipos se lo pasan entre sí.

La figura 13.7 modela una red ARCnet con un concentrador pasivo, uno activo y varios equipos.

**FIGURA 13.7**

*Un diagrama de distribución de una red ARCnet.*




www.elsolucionario.net

## Thin ethernet

TÉRMINO NUEVO

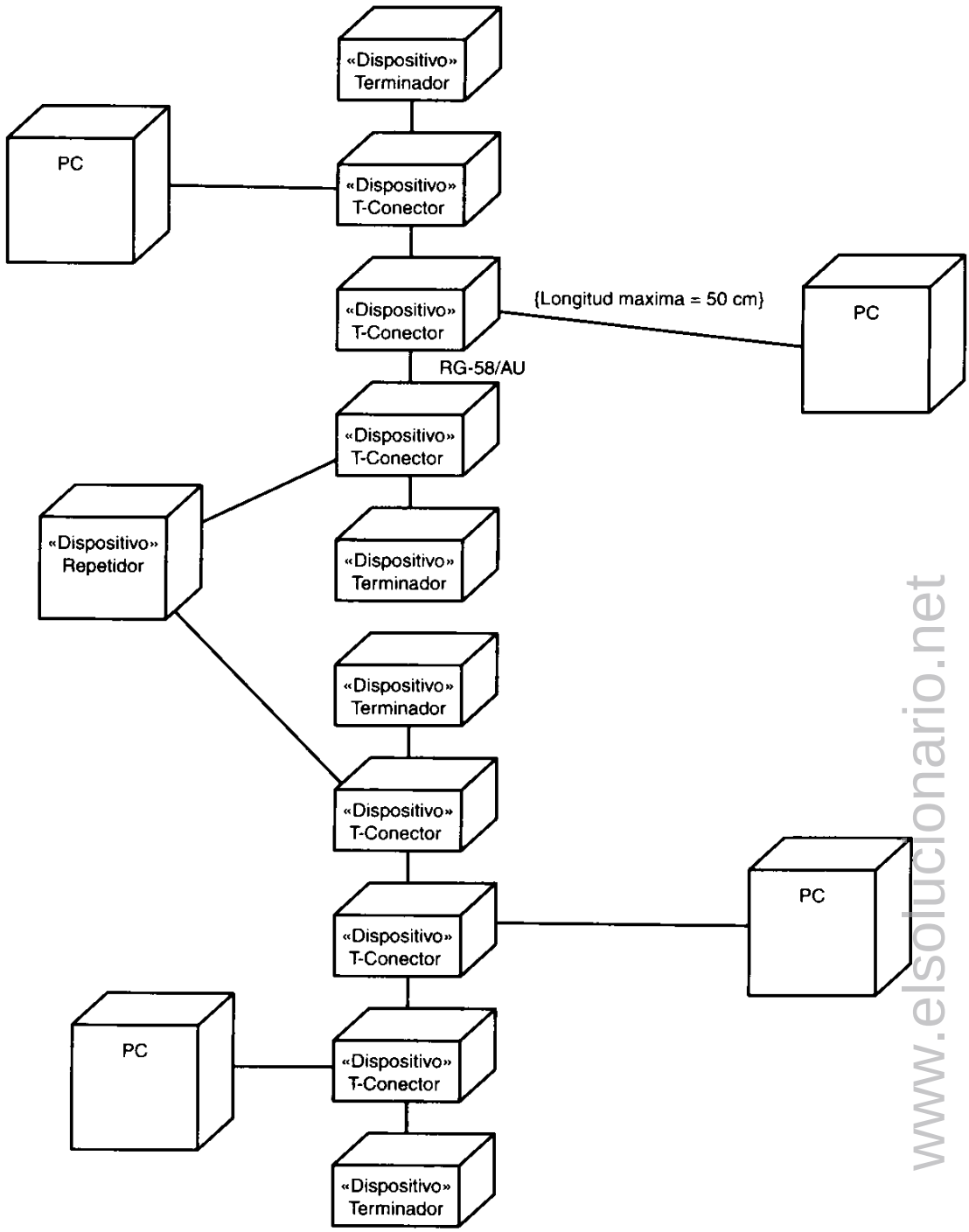
La red thin ethernet es un tipo muy popular. Los equipos se conectan a un cable de red mediante dispositivos conocidos como conectores T. Un segmento de red puede unirse a otro mediante un *repetidor*, un dispositivo que amplifica una señal antes de transmitirla. También pueden hacerse conexiones de tipo RJ-45, aunque, en este caso, nos concentraremos tan sólo en la conexión T.

La figura 13.8 modela una red thin ethernet.

www.elsolucionario.net

**FIGURA 13.8**

*Diagrama de distribución de una red thin ethernet.*




www.elsolucionario.net

## Red inalámbrica Ricochet de Metricom

Metricom, Inc, empresa localizada en Los Gatos, CA, cuenta con una solución inalámbrica por módem para obtener acceso móvil a Internet. Su módem inalámbrico se conecta al puerto serial de un equipo de cómputo y se comunica con su red Ricochet.

La red Ricochet consta de transmisores y receptores de radio, cuyo tamaño es de una caja de zapatos. Tales radios de microceldilla se montan en la parte superior de los postes de luz a distancias de 400 a 800 metros, en un patrón de tablero de ajedrez. Cada radio de microceldilla obtiene una pequeña cantidad de energía de su poste de luz si se equipa con un adaptador especial.

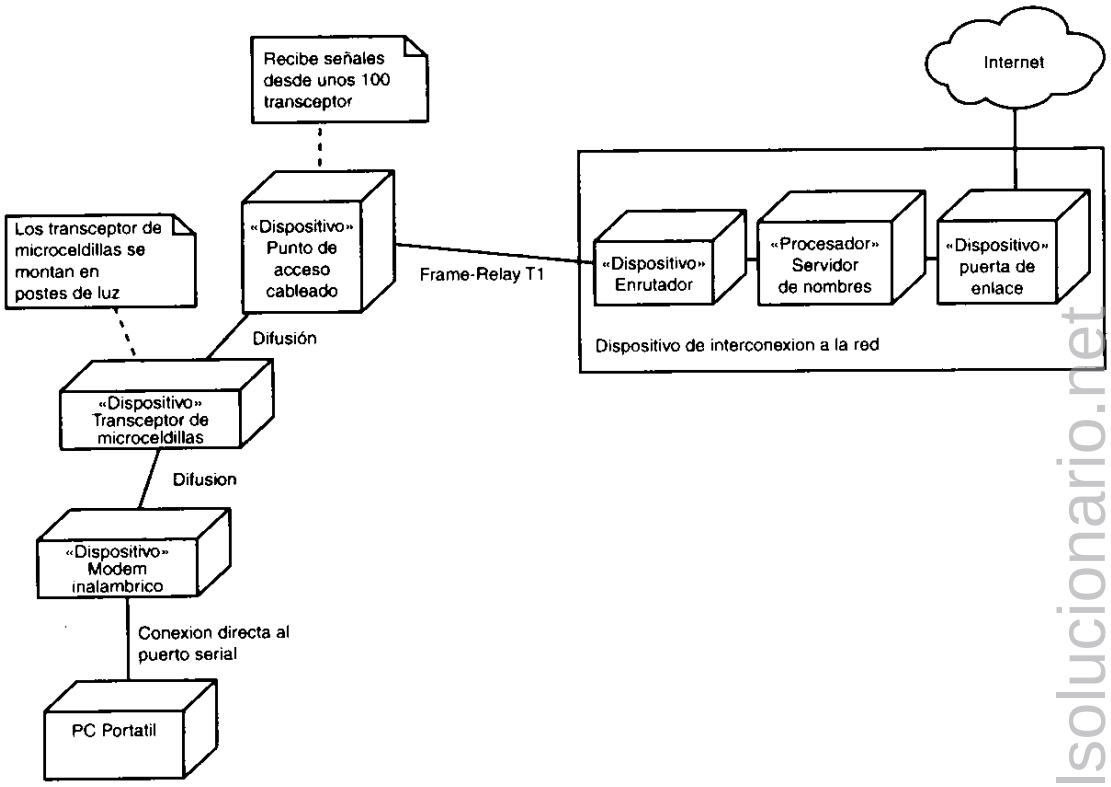
www.elsolucionario.net

Los radios de microceldillas difunden señales a Puntos de acceso cableados que llevan la información a un NIF (dispositivo de interconexión a la red). El NIF consta de un servidor de nombres (una base de datos que valida las conexiones), un enrutador (dispositivo que enlaza a las redes entre sí), y una puerta de enlace (un dispositivo que traduce la información de un protocolo de comunicaciones a otro). La información se lleva del NIF a la Internet.

La figura 13.9 muestra un diagrama de distribución para esta red.

**FIGURA 13.9**

*Red inalámbrica Ricochet de Metricom.*




# Los diagramas de distribución en el panorama

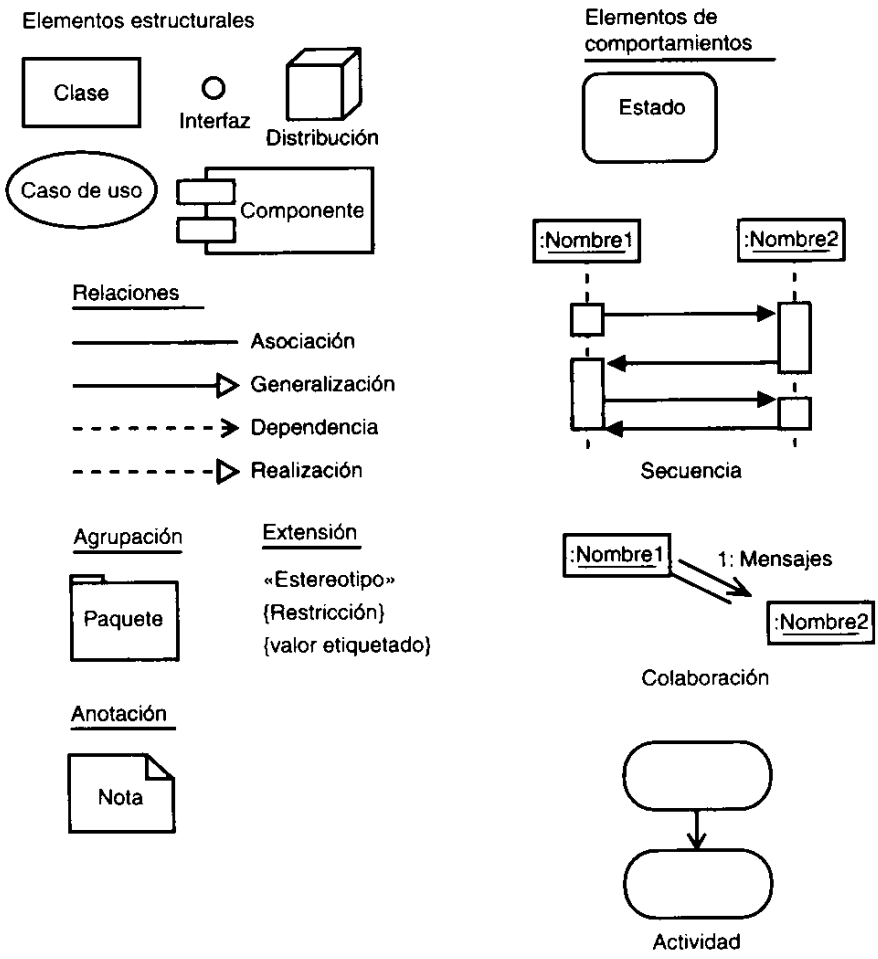
Ha llegado al final del conjunto de diagramas. El panorama incluye al diagrama de distribución, y ha quedado finalizado.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 13.10**

*Su panorama del UML incluye al diagrama de distribución y está completo.*




# Resumen

El diagrama de distribución del UML ilustra la forma en que luce un sistema físicamente cuando sea conjugado. Un sistema consta de nodos, donde cada nodo se representa por un cubo. Una línea asocia a dos cubos y simboliza una conexión entre ellos. Los tipos de nodos son procesador (que puede ejecutar un componente) y dispositivo (que no lo puede hacer). Los dispositivos por lo general interactúan con el mundo.

Como puede imaginar, los diagramas de distribución son útiles para modelar redes. Los modelos presentados en esta hora incluyeron a redes token-ring, ARCnet, thin ethernet y la red inalámbrica Ricochet.

www.elsolucionario.net

www.elsolucionario.net

# Preguntas y respuestas

**P** **Usted utilizó una nube para representar a la Internet, y dijo que no era parte de la simbología del UML. ¿Un modelador puede utilizar símbolos que no están en la simbología?**

**R** Así es. De hacerlo, no habrá policía UML que lo lleve a prisión. La idea es utilizar el UML para expresar una visión. En ninguna parte esto es tan útil como en los diagramas de distribución. Si tiene una imagen que pueda mostrar claramente los equipos de escritorio, portátiles, servidores y otros procesadores (o dispositivos), podrá utilizarlos en sus diagramas. Claro que estará creando un estereotipo gráfico. (Por cierto que el símbolo de la nube es una interesante nota al margen por aprender en el UML. Uno de los creadores del UML, Grady Booch, solía representar objetos como nubes en la simbología de su esquema de modelado antes de que se convirtiera en parte del equipo del UML.)

**P** **Suponga que cuenta con una gran cantidad de figuras para representar a ciertos objetos y no a otros. ¿Se pueden mezclar con los símbolos del UML?**

**R** Claro que puede. El objeto es dibujar diagramas para clarificar una visión, no para (perdón por el juego de palabras) nublarla.

# Taller

Ahora que ha finalizado con todo el conjunto de diagramas del UML, pruebe su conocimiento respecto a la forma de representar hardware. Las respuestas se destacan en el apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿Cómo representa a un nodo en un diagrama de distribución?
2. ¿Qué tipo de información puede aparecer en un nodo?
3. ¿Cuáles son los dos tipos de nodos?
4. ¿De qué forma funciona una red token-ring?

## Ejercicios

1. Imagine a su equipo de cómputo doméstico como un conjunto de nodos. Dibuje un diagrama de distribución que incluya a su gabinete, monitor, impresora y cualquier otro periférico. Incluya cualquier estereotipo necesario y compartimientos para clarificar la información (posiblemente resultará en un modelo algo diferente al de mi equipo).
2. Es posible conectar una red a otra. Una forma de hacerlo es conectar a cada red con un enrutador y a cada uno de ellos con un (posiblemente muy grande) circuito de LAN a LAN. Dibuje un diagrama de distribución de una pequeña red token-ring que se conecte a una pequeña red thin ethernet.

www.elsolucionario.net

www.elsolucionario.net

# HORA 14

# Nociones de los fundamentos del UML

Ahora que ha visto los diagramas en el UML, ya está listo para ciertos conceptos fundamentales.

En esta hora se tratarán los siguientes temas:

- Estructura del UML
- Capa del metamodelado
- Extensión del UML
- Estereotipos, restricciones y valores etiquetados

Si éste fuera un texto académico en lugar de uno de autoaprendizaje, esta hora hubiese aparecido al principio de la parte I, en lugar de estar casi al final. Hice esto para darle la oportunidad de conocer las trincheras del UML (qué es y lo que hace). Así, ya estará listo para conocer los fundamentos y trabajar con ellos.

Ahora que ha visto los diagramas y sabe cómo utilizarlos, ¿para qué podría servirle esta hora? Si ya comprendió en qué se basa el UML, podrá extenderlo y adaptarlo cuando empiece a utilizarlo en el mundo real. Como cualquier analista de sistemas le diría: cada proyecto es diferente. No existe ningún texto de referencia, libro o manual que lo pueda preparar para todas las situaciones con las que se encontrará. No obstante, el tener una buena base de los conceptos fundamentales lo preparará para la mayoría de los sistemas que tenga que modelar.

www.elsolucionario.net

www.elsolucionario.net

Es como aprender un idioma extranjero. La mejor forma de hacerlo es sumergirse en él, como lo hizo en las horas 1 a la 13 (y como lo hará en la parte II, "Estudio de un caso"). Luego podrá empezar a captar las reglas gramaticales y sintácticas dado que estará preparado para comprenderlas (¡por desgracia, muchos cursos académicos de idiomas proceden en orden opuesto!).

# Estructura del UML

Su panorama del UML le muestra las categorías de los diagramas y a éstos en cada categoría. Como lo dije en la hora 1, "Introducción al UML", necesitará todos los diagramas, ya que le permitirán ver su sistema desde diversos puntos. Debido a que hay varias personas a las que les interesa el sistema por distintas razones, deberá tener la capacidad de comunicar una visión consistente del sistema de diversas formas.

Aunque su panorama es útil como una forma de tener a la mano los elementos del UML, no funciona como una definición de éste. Los tres amigos estructuraron al UML de una manera formal para asegurarse que los elementos que habían creado pudieran mostrar una idea clara de un sistema propuesto, o completamente reestructurado.

El UML cuenta con una arquitectura de cuatro capas. Tales capas se distinguen por la generalidad de los elementos que en ellas residen.

En la actualidad, la palabra *arquitectura* nos brinca en el mundo del desarrollo de sistemas. Imagine una arquitectura como una forma de resumir un conjunto de decisiones respecto a la forma en que se organiza un sistema.
Tales decisiones se enfocan muy específicamente en los elementos del sistema: qué son, qué hacen, cómo se comportan, cómo se relacionan y se combinan.

En las horas anteriores estuvo operando en las dos capas más específicas. Cuando siguió un ejemplo o realizó un ejercicio que involucraba instancias específicas de un dominio en particular (como el diagrama de componentes de mi sistema de cómputo personal), se encontraba en la capa más específica. Esta capa se llama *capa de objetos del usuario*, donde "usuario" se refiere a quien utiliza el UML (no al que utiliza el sistema en sí).

TERMINO NUEVO

Estuvo en la siguiente capa cuando vio las clases, como en el ejemplo donde el analista habló con el entrenador de baloncesto para distinguir las clases en el dominio baloncesto. Los primeros estados del análisis tienen que ver con esta capa: trabajará con un experto o un cliente para obtener la información de un dominio, y con los usuarios potenciales para comprender los casos de uso que tendrán que tomarse en cuenta al generar el sistema. A esta capa se le denomina *capa de modelado*.

www.elsolucionario.net

www.elsolucionario.net

**TÉRMINO NUEVO** ¿En qué momento estuvo en una capa menor? Al principio de cada hora cuando aprendió un concepto como el de una clase o un nodo, estaba en la tercera de cuatro capas. Ésta define al lenguaje para un modelo específico. Luego de que obtenga algo de experiencia, se familiarizará tanto con el UML que esta tercera capa le será algo muy natural. Dado que esta capa define lo que va en un modelo, se conoce como *capa de metamodelado*.

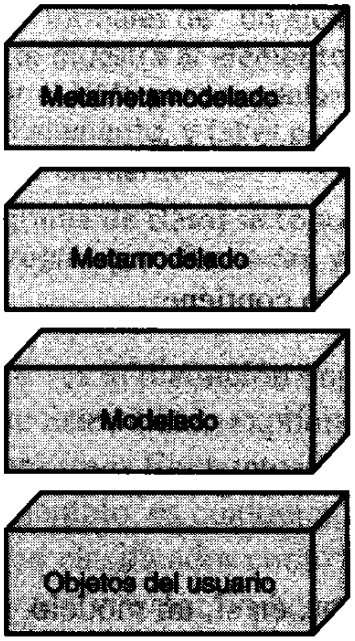
Puesto que en su panorama se muestran los símbolos de las clases, nodos, componentes, casos de uso y cosas así, tal panorama pertenece a la capa de metamodelado.

**TÉRMINO NUEVO** ¿Y la cuarta capa? Durante su carrera como analista, posiblemente nunca tendrá que tratar con ella. Imagínela como una forma de definir un lenguaje que especifique clases, casos de uso, componentes y todos los demás elementos del UML con los que trabajará. Esto es más de la incumbencia de los teóricos que diseñan y comparan lenguajes que de los analistas que lo usan. Dado que esta capa define lo que va en el metamodelo, se conoce como *capa de metametamodelado.*

La figura 14.1 le muestra las cuatro capas.

**FIGURA 14.1**

*Las cuatro capas del UML.*




# Capa del metamodelado: cercano y personal

Puesto que la capa de metamodelado es la base de su panorama, examinémosla más detalladamente.

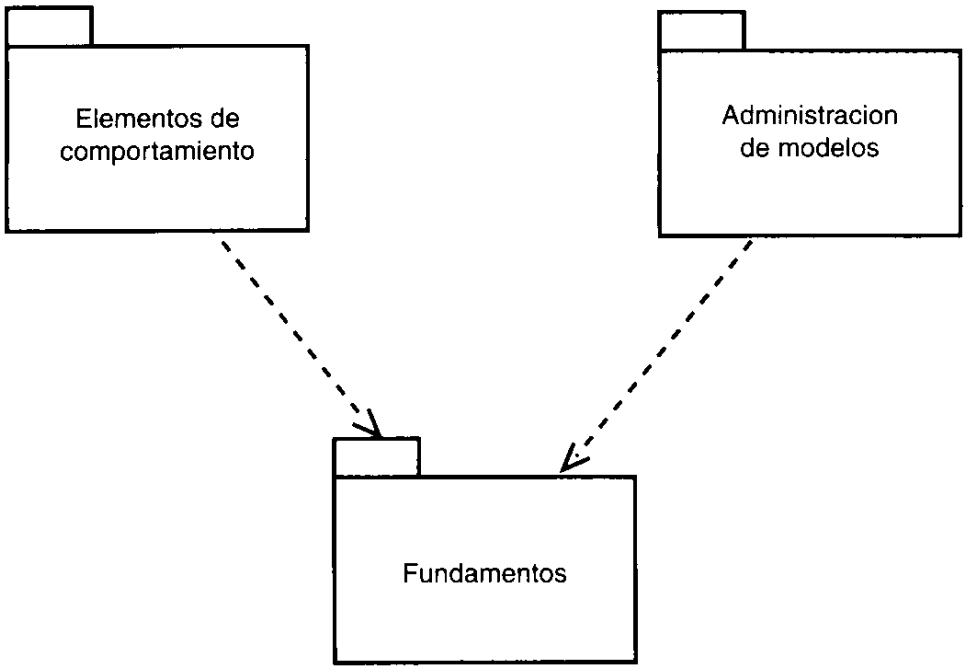
www.elsolucionario.net

www.elsolucionario.net

Imagine a esta capa como una formación de tres paquetes: *Fundamentos, Elementos de comportamiento* y *Administración de modelos*. La figura 14.2 le muestra lo que quiero decir.

**FIGURA 14.2**

*Los paquetes en la capa de metamodelado del UML.*




Como en el caso de cualquier paquete, cada uno de estos grupos relacionan elementos entre sí. (¿Utilizamos al UML para modelar al UML? ¡Por supuesto!)

¿Cuáles son estos elementos? El paquete Fundamentos contiene:

- Núcleo
- Elementos auxiliares
- Tipos de datos
- Mecanismos de extensión

El paquete de Elementos de comportamiento contiene:

- Comportamiento en común
- Colaboraciones
- Casos de uso
- Máquinas de estado

El paquete de Administración de modelos es, en sí, un modelo. Es un diagrama de clases que muestra cómo se relacionan los elementos del UML entre sí, como son los paquetes o subsistemas.

## El paquete de Fundamentos

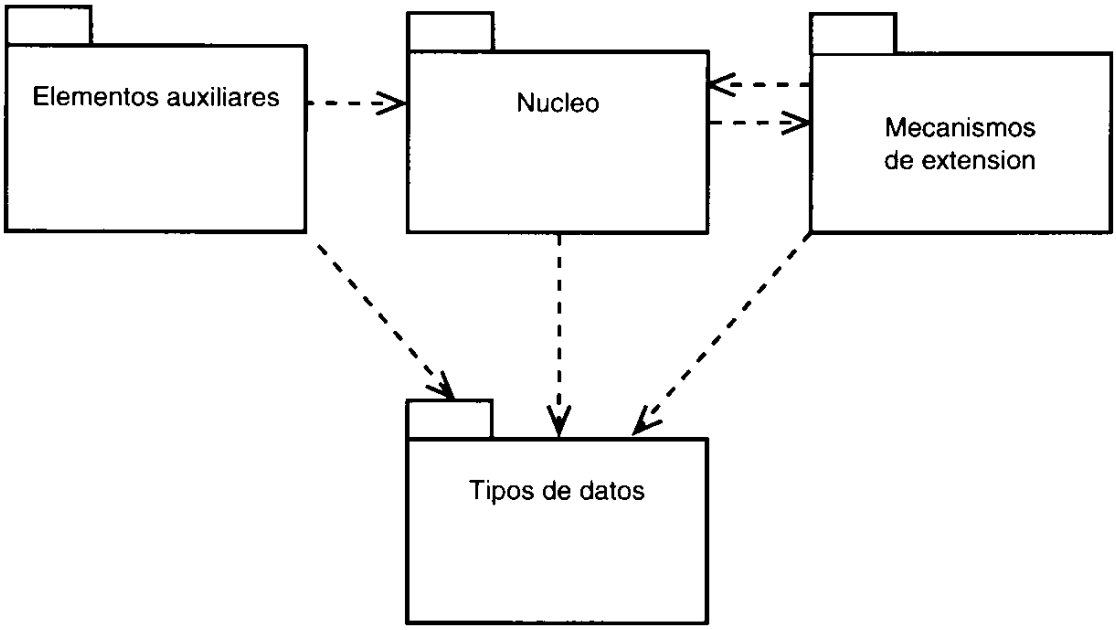
Hagamos una retrospectiva y entremos a otro nivel. Empezaré con el paquete de fundamentos, cuyos componentes aparecen en la figura 14.3.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 14.3**

*Los paquetes del propio paquete de Fundamentos.*




El núcleo define lo que necesita para crear un modelo UML. Cada uno de los elementos definidos es *abstracto* (lo que significa que no puede crear instancias de él) o *concreto* (con el que sí podrá). Entre los elementos abstractos se encuentran *ElementoDeModelo, ElementoGeneralizable y Clasificador*. Entre los concretos se encuentran *Clase, Interfaz, Asociación y Tipo de datos*.

Verá en esta sección que diré que un paquete "define", "establece" o "da los detalles formales de" un elemento (o concepto). Ello significa tres cosas: (1) el paquete muestra al elemento dentro de un diagrama de clases (otro ejemplo de usar al UML para modelar al UML), (2) el paquete contiene reglas para utilizar el elemento, y (3) el paquete proporciona información respecto al significado del elemento.

El diagrama de clases se conoce como *sintaxis abstracta*, las reglas se conocen como *reglas bien formadas*, y al significado se le conoce como *semántica*.

He aquí otra forma de ver la distinción entre abstracto y concreto. Nunca tendrá nada en su modelo a lo que usted llame explícitamente ElementoDeModelo o Clasificador (aunque claro, utilizará *tipos* ElementosDeModelo y Clasificadores todo el tiempo). Un clasificador, por ejemplo, es cualquier elemento que describe estructura y comportamiento. Imagine a un clasificador como una forma resumida de referirse a una *clase, componente, nodo, actor, interfaz, indicación, subsistema, caso de uso o tipo de datos*. Decir que algo se le aplica a un clasificador es equivalente a decir que se aplica a cualquiera de estos otros denominadores.

En tal caso, ¿los elementos concretos se derivan de los abstractos? Así es. ¿Ello significará que hablamos de clases y herencia? Por supuesto, pero dado que estamos en la capa de metamodelado, en realidad hablamos de *metaclases*. Por ello, un "clasificador" es una "metaclase". (¿Qué tanto sentido podría haber tenido esta frase en la hora 1?)

www.elsolucionario.net

www.elsolucionario.net

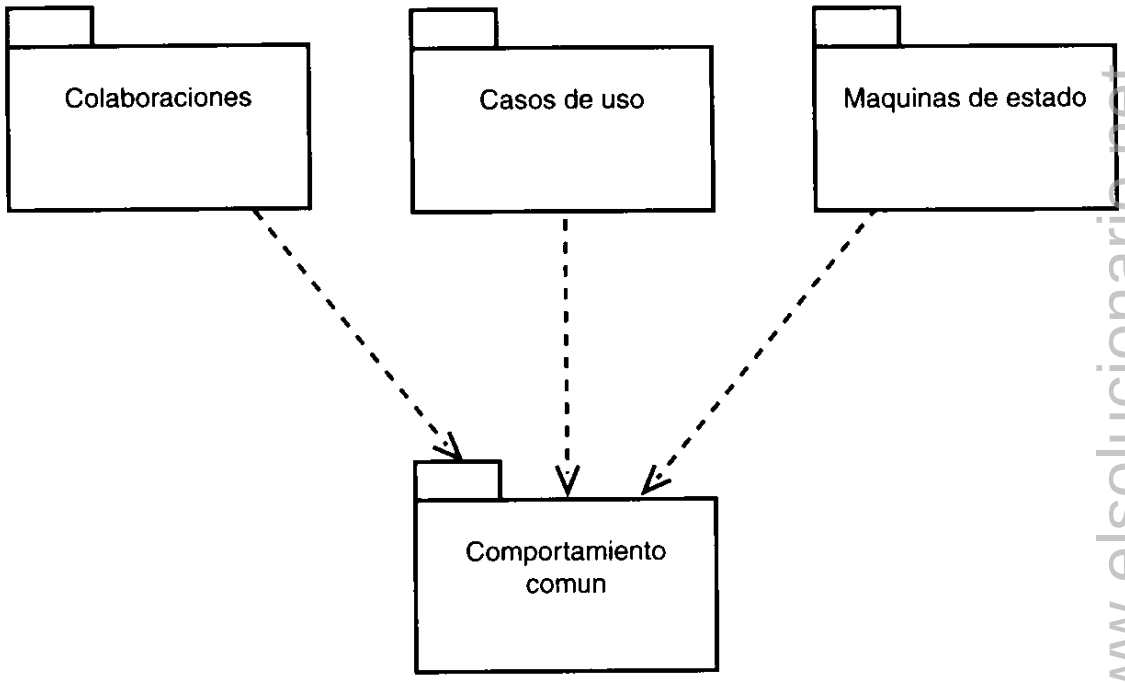
Continuemos con los otros paquetes dentro de los fundamentos. Elementos Auxiliares, un paquete que redeondea al Núcleo, define la Dependencia, Componentes y Nodos, entre otros. El paquete Tipo de Datos, especifíca los tipos de datos que el UML utiliza, incluyendo los tipos primitivos (enteros, cadena y tiempo) y enumeraciones. Una enumeración como por ejemplo el Booleano lista los valores posible. El paquete de Mecanismos de Extensión le especifica cómo puede extender el UML e incluye algunas extensiones ya hechas. Analizaremos de manera más detallada este paquete más adelante, en la sección llamada Extensión del UML.

## El paquete de los elementos de comportamiento

Este paquete es la parte del UML que se encarga de modelar el procedimiento de un sistema. Los paquetes de que consta aparecen en la figura 14.4.

**FIGURA 14.4**

*Los paquetes que conforman al propio paquete de Elementos de comportamiento.*




El paquete de comportamiento común proporciona los conceptos de los elementos dinámicos, y soporta otros paquetes como son: casos de uso, máquinas de estado y colaboraciones. Estos "conceptos" incluyen *Señal*, *Enlace* y *Punto final de asociación*.

El paquete de colaboraciones abarca un ámbito más amplio que tan sólo los diagramas de colaboraciones que utilizó en la hora 10, "Diagramas de colaboraciones". En este contexto, una "colaboración" describe la forma en que los clasificadores y sus asociaciones trabajan en conjunto para realizar una tarea en particular. Los diagramas de colaboraciones y los de caso de uso son parte de la perspectiva. La idea es que los clasificadores conformen al *contexto* de la colaboración; las asociaciones conforman a la *interacción*.

No es de extrañar que el paquete de casos de uso detalle los conceptos (como a un *Actor* y un *CasoDeUso*) que en él se encuentran. (Ambos conceptos son clasificadores, como lo indiqué en la sección anterior.) El objetivo general es tener la posibilidad de describir el comportamiento de un sistema sin entrar en detalles.

www.elsolucionario.net

www.elsolucionario.net

Tampoco debe sorprender que el paquete de Máquinas de estado dé los detalles formales de los conceptos que hay detrás de los diagramas de estados y de actividad que ya ha utilizado.

### Administración de modelos

Este paquete define al *Modelo*, *Subsistema* y *Paquete*. La meta de estos elementos es agrupar los *ElementosDeModelo* de todo tipo.

## Extensión del UML

Como ya lo ha visto en las horas anteriores, podrá pulir sus diagramas UML mediante la adición de detalles que expliquen mejor su significado. Los estereotipos, restricciones y valores etiquetados son herramientas útiles dentro del UML.

Puede crear una extensión sobre la marcha para agregar a su modelo cuestiones e ideas importantes de su dominio. Esto fue evidente la hora anterior: las restricciones en algunas de las comunicaciones entre nodos revelan las reglas de los diversos tipos de redes.

El UML incluye varios estereotipos, restricciones y valores etiquetados. Como ya lo mencioné, son parte del paquete Extensiones el cual, a su vez, se encuentra en el paquete Fundamentos de la capa de metamodelado. Cada una de estas extensiones incluidas es adecuada para uno (a veces dos) de los elementos del UML. Las siguientes secciones tratarán a tales extensiones.

## Estereotipos

El propósito de un estereotipo es extender a un elemento del UML para que sea una instancia de una nueva metaclase, y se escribe entre dos pares de paréntesis angulares. Esto agrega una gran flexibilidad. Lo que significa que usted podrá utilizar un elemento existente del UML como base para crear sus propios elementos: elementos que capturen algún aspecto de su propio sistema o dominio de una forma en que no podrían hacerlo los elementos del UML.

La intención del estereotipo es permitir a la entidad recién creada que embone con los demás dentro de una herramienta de modelado. Las herramientas de modelado (como Rational Rose, SELECT Enterprise o Visual UML) tienen que almacenar y manejar las clases para la generación de código y la de informes. El mecanismo del estereotipo les permite hacerlo con sus creaciones.

El UML incluye un extenso conjunto de estereotipos generados. Podrá agregar de uno en uno o dos elementos. Las siguientes subsecciones organizan a los estereotipos en términos de los elementos con que confluyen.

www.elsolucionario.net

www.elsolucionario.net

## Dependencia

La relación de dependencia puede tomar la mayor cantidad de estereotipos ya creados. Cada uno extiende una relación de dependencia entre un origen (el elemento del cual parte la flecha punteada) y un destino (el elemento al que apunta la flecha). Veamos rápidamente a cada dependencia estereotipada.

Una dependencia de tipo «se convierte en» muestra que el origen y el destino son el mismo objeto en distintos momentos. El origen se convierte en el destino con (posiblemente) diferentes roles y valores. «llamar» tiene una operación como origen y otra como su destino. En esta dependencia estereotipada, la operación de origen invoca a la de destino. Una dependencia «copiar» indica que el destino es una copia exacta del origen. En una dependencia «derivar», el origen se deriva del destino.

¿Recuerda el concepto de visibilidad de la hora 5, "Agregación, composición, interfaces y realización"? Si tiene una operación que sea privada dentro de una clase en particular, aún podrá hacerla accesible a otra clase. Coloque la otra clase (origen) y la operación (destino) en una dependencia «reunir» o «friend». El origen tendrá acceso al destino sin importar la visibilidad.

Una dependencia entre dos casos de uso también puede tener un estereotipo. Ya ha utilizado dos de ellos, «extender» y «usar», aunque sustituyó «incluir» por «usar». «extender» le indica que los comportamientos del caso de uso de destino se agregan al caso de uso de origen. «usar» indica que algunos casos de uso tienen cierto comportamiento en común, y este estereotipo le permite utilizar dicho comportamiento sin tener que repetirlo una y otra vez.

Una dependencia «importar» se establece entre dos paquetes. Este estereotipo agrega el contenido del destino al espacio de nombres del origen (el aspecto del paquete que agrupa los nombres que lo constituyen).

El estereotipo «instancia» indica que el origen es una instancia de su destino, que siempre será un clasificador. En una dependencia de «metadestino», tanto el destino como el origen son clasificadores, y el destino es la metaclase del origen.

En un «enviar», el origen es una operación y el destino es una señal. El estereotipo muestra que el origen envía la señal.

## Clasificador

Los estereotipos extienden a los clasificadores de diversas formas. El estereotipo «metaclase» muestra que el clasificador al que está adjunto es una metaclase de otra clase. El «tipodeautoridad» indica que un clasificador tiene objetos que provienen de un antecesor en particular. También puede usar «tipodeautoridad» en una dependencia para mostrar que el destino es un tipo de potestad del origen. (Normalmente utilizará éste cuando modele bases de datos.)

www.elsolucionario.net

www.elsolucionario.net

Los estereotipos «proceso» y «subproceso» tienen que ver con el flujo de control. Ambos indican que su clasificador es una clase activa; es decir, sus objetos pueden iniciar la actividad de control. Un proceso puede consistir de varios subprocesos (flujos de control), y puede ejecutarse al mismo tiempo que otros procesos. Un subproceso puede ejecutarse junto con otros subprocesos en el mismo proceso.

Un clasificador con el estereotipo «utileria» es una colección titulada de atributos y operaciones que no son miembros de tal clasificador: un clasificador que no tiene instancias.

Finalmente, un clasificador puede tener al abuelo (¡casi literal!) de todos los estereotipos: «estereotipo». Este indica que el clasificador funciona como un estereotipo, y le permite modelar jerarquías de estereotipos.

## Clase

Puede obtener algo más específico que con los clasificadores: también es posible extender a una clase. Un «tipo» es una clase que establece un dominio de objetos junto con atributos, operaciones y asociaciones. El «tipo» no contiene métodos (algoritmos ejecutables para sus operaciones).

Una «claseDeImplementacion» es lo contrario de un «tipo». Representa la implementación de una clase en un lenguaje de programación.

## Generalización

Es una relación entre clasificadores, con su propio pequeño conjunto de estereotipos. «heredar» significa que las instancias del subtipo no pueden substituirse por instancias del supertipo. «subclase» hace lo mismo que en las clases: significa que las instancias de la subclase no son sustituibles por instancias de la superclase. «privado» denota una herencia exclusiva: oculta los atributos heredados y operaciones de una clase a sus ancestros.

## Paquete

Los estereotipos de los paquetes son directos. Una «fachada» es un paquete que contiene referencias a elementos de otro paquete, y que no contiene elementos propios. Un «sistema» es una colección de modelos de un sistema. Un «cabo» es un paquete que proporciona sólo las partes públicas de otro paquete.

**Término Nuevo**

Además de los elementos que he dicho, un paquete puede incluir patrones. Un patrón es un tipo de colaboración entre los elementos que ha probado su efectividad en diversas situaciones. Un «marcoDeTrabajo» es un paquete estereotipado que sólo contiene patrones.

www.elsolucionario.net

www.elsolucionario.net

Dado que los paquetes pueden encontrarse dentro de paquetes, sirve de mucho tener un estereotipo que indique cuál de los paquetes está en el nivel superior. Tal estereotipo es el «paqueteDeNivelSuperior». (¡Le dije que los estereotipos de los paquetes son directos!)

## Componente

Los estereotipos para los componentes son aún más directos. Puede mostrar que un componente es un documento, un ejecutable, un archivo, una tabla de datos o una biblioteca. Los estereotipos respectivos son «documento», «ejecutable», «archivo», «tabla» y «biblioteca».

## Algunos otros estereotipos

En esta subsección, he conjugado algunos estereotipos utilizados con otros elementos del UML.

Un comentario que aparece en una nota adjunta puede tener un estereotipo «requerimiento», que denota que el comentario establece un requisito para el elemento adjunto a la nota.

Dentro de una clase, una operación o un método pueden crear una instancia o destruirla. (Quizá haya visto métodos como *constructor* y *destructor* en Java.) Tales características se indican mediante «crear» y «destruir» respectivamente.

Las restricciones, que son el mecanismo de extensión que ahora trataré, pueden también funcionar con los estereotipos. En ocasiones usará una restricción para mostrar las condiciones previas a una operación; aunque algunas veces mostrará sus condiciones posteriores. Tales restricciones las estereotipará con «condicionPrevia» o «condicionPosterior». En ocasiones adjuntará una restricción a un conjunto de clasificadores o relaciones, y necesitará indicar que las condiciones de la restricción deberán tener todos los clasificadores, relaciones e instancias. Para ello, deberá estereotipar la restricción como «invariable».

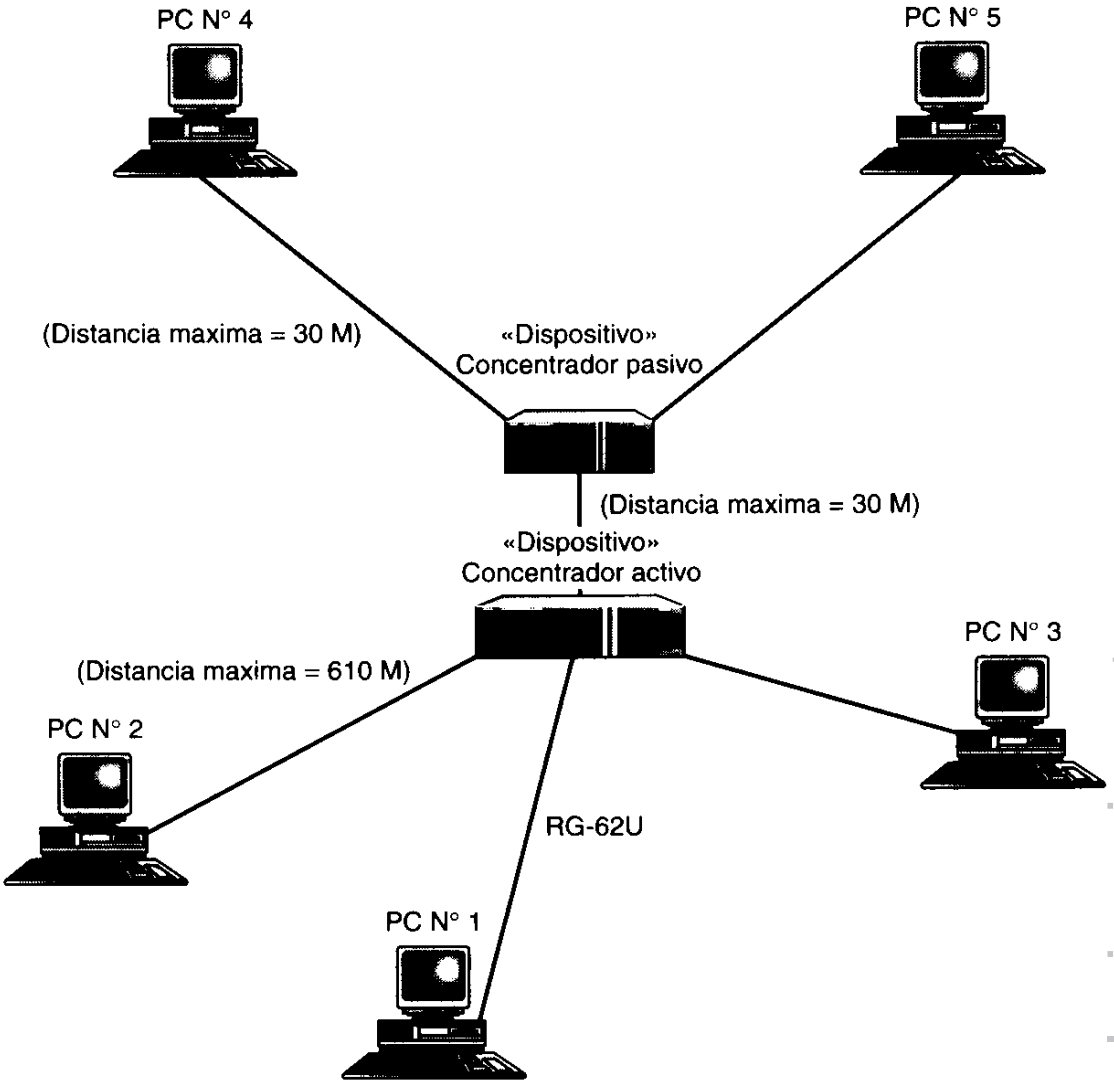
## Estereotipos gráficos

Algunas veces en su dominio, tal vez tendrá que obtener un nuevo símbolo o dos para transmitir algo a un cliente. Como lo mencioné en la hora anterior, los diagramas de distribución pueden encajar de forma importante en esto. Por lo general, hay disponibles figuras de procesadores y dispositivos, y pueden remplazar a los cubos que vio en la hora 13, "Diagramas de distribución". La figura 14.5 le muestra un ejemplo. Es una versión estilizada de la figura 13.7, un modelo de una ARCNet.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 14.5**
*Un modelo estilizado de una ARCNet.*




# Restricciones

Las restricciones se encuentran entre llaves. Proporcionan las condiciones para las asociaciones, extremos de vínculos, generalizaciones y peticiones (transmisiones de señales o llamadas a operaciones).

La restricción {o} se aplica a un conjunto de asociaciones, y muestra que sólo una asociación podrá utilizarse para tal conjunto. Otra restricción basada en asociaciones, {implícito}, indica que una asociación es conceptual.

Los extremos de vínculos, que son puntos finales de vínculos entre objetos, pueden contener cualquier cantidad de restricciones. Cada restricción indica el porqué el objeto en el extremo del vínculo es visible. {parámetro} muestra que el objeto es un valor necesario relativo al vínculo. {propio} le indica que el objeto es el despachador de una petición. {global} y {local} indican el ámbito del objeto respecto al vínculo. {asociación} denota que el objeto es visible por su coalición.

www.elsolucionario.net

www.elsolucionario.net

Un conjunto de generalizaciones pueden ser {completo} (todos sus subtipos han sido especificados) o {incompleto} (aún pueden agregarse subtipos). Otro conjunto de generalizaciones pueden ser {traslapado} (más de un subtipo puede funcionar como un tipo de instancia) o {desarticulado} (sólo un subtipo puede ser un tipo de una instancia, lo que es predeterminado en la generalización).

Si una petición se envía a diversas instancias de destino en un orden no especificado, es una {difusión}. Si varias instancias devuelven valores, y la mayoría de tales valores determinan un solo valor, la restricción será un {voto}.

# Valores etiquetados

Un valor etiquetado se escribe entre llaves. Consiste en una *etiqueta*, un signo = y un *valor*.

Los valores etiquetados ya elaborados son adecuados para los clasificadores, componentes, atributos, instancias y operaciones. Una etiqueta {documentación = }, se aplica a cualquier elemento. A la derecha del =, debe colocar una descripción, explicación o comentario respecto al elemento al que adjuntó este valor etiquetado.

Puede adjuntar una {ubicación = } a un clasificador o componente. Cuando lo adjunta a un clasificador, debe proporcionar al componente del cual es parte. Cuando lo hace a un componente, debe indicar el nodo donde se encuentra.

El valor etiquetado {persistencia = } puede ir en un atributo, clasificador o instancia. Denota la permanencia del estado del elemento al que lo ha adjuntado. Los valores posibles son *permanente* (el estado persiste cuando la instancia se destruye) y *transitorio* (cuando la instancia se destruye, también lo hace el estado).

{semántica = } especifica el significado de un clasificador o una operación. {responsabilidad} es una obligación de un clasificador.

# Resumen

Esta hora trató los conceptos básicos del UML. El objetivo fue el de darle un conocimiento profundo que le permitirán aplicar al UML en situaciones reales que no siempre puedan reflejarse en los ejercicios de un libro de texto. Hemos tratado estos conceptos luego de todos los diagramas porque tenía que comprender los elementos del lenguaje antes de profundizar en sus fundamentos.

El UML consta de cuatro capas: objetos del usuario, modelado, metamodelado y metametamodelado (que van desde específicos hasta generales). Cuando analice un sistema, típicamente trabajará en las primeras dos capas. Cuando aprenda los conceptos del UML, por lo general se encontrará en la tercera. La cuarta capa está orientada a los teóricos y diseñadores del lenguaje, en lugar de los usuarios del lenguaje y analistas de sistemas.

www.elsolucionario.net

www.elsolucionario.net

El UML incluye varias extensiones propias. Cada uno de estos estereotipos, restricciones y valores etiquetados se orientan a ser usados por uno o dos de los símbolos del UML.

Si le hubiera dado todos estos conceptos fundamentales al iniciar en la hora 1 ¿los habría entendido?

# Preguntas y respuestas

**P Veo que el UML tiene varias reglas. ¿Quién las implementa?**

**R** Como ya dije, la policía UML no le acecha para verificar que su modelo sea preciso. No obstante, una herramienta de modelado le podrá ayudar a ceñirse a las reglas. Por ejemplo: Visual UML tiene un archivo de estereotipos que usará de un modo sensible al contexto. Cuando intente colocar un estereotipo en un elemento en particular, sólo le permitirá seleccionar de los estereotipos adecuados para tal elemento, todos ellos en inglés. Además, le permitirá agregar estereotipos a su archivo de estereotipos.

**P Los valores etiquetados parecen ser esotéricos. ¿Debo utilizarlos?**

**R** Sí, y con frecuencia. Los valores etiquetados integrados, aunque son útiles, palidecen en comparación con los que usted definirá para sí. Puede utilizar un valor etiquetado para mantener información relacionada con la administración de un importante proyecto en su modelo: como los números de versión y los autores de las clases. Es decir, "estado", "versión" o "autor" podrían ser sus etiquetas, y usted establecerá los valores adecuados.

# Taller

Este taller reafirmará su conocimiento de los fundamentos del UML. Utilice sus procesos de reflexión en el cuestionario y localice las respuestas en el Apéndice A, "Cuestionario". Ésta es una hora teórica, por lo que no he incluido ningún ejercicio.

## Cuestionario

1. ¿Cuáles son las cuatro capas del UML?
2. ¿Qué es un clasificador?
3. ¿Porqué es importante el poder extender al UML?
4. ¿Cuáles son los mecanismos de extensión del UML?

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 15

# Adaptación del UML en un proceso de desarrollo

Ahora que ha comprendido los diagramas del UML y su estructura, ya casi es hora de ponerlo a funcionar. El UML es una herramienta maravillosa, pero no la utilizará de manera aislada. El UML tiene la intención de impulsar el desarrollo de software, por ello, es necesario aprender los procesos y metodologías de desarrollo como un vehículo para comprender el uso del UML en un contexto.

En esta hora se tratarán los siguientes temas:

- Por qué es importante un proceso de desarrollo
- Por qué no son adecuadas las antiguas metodologías para los sistemas de hoy
- El Proceso de desarrollo GRAPPLE
- Cómo incorporar al UML en el proceso

Su empresa requiere un nuevo sistema de cómputo. Los nuevos componentes de hardware y software darán por resultado una ventaja competitiva, misma que usted necesitará. Debe empezar el desarrollo, ¡y pronto!

www.elsolucionario.net

www.elsolucionario.net

Es usted quien ha tomado la decisión de generar al nuevo sistema, ha establecido un equipo de desarrollo completo, con un gerente de proyectos, modeladores, analistas, programadores e ingenieros de sistemas. Ya todos se truenan los dedos, ansiosos por empezar.

Usted es, en otras palabras, un cliente. ¿Cuáles productos de trabajo esperará obtener del equipo de desarrollo? ¿Cómo querrá que el gerente de proyectos le reporte? Al final, claro, querrá un sistema en funcionamiento; pero antes de ello, necesitará ver indicios de que el equipo ha comprendido el problema que usted intenta resolver y su visión para hacerlo. Necesitará ver el avance de su solución, y necesitará saber cuál ha sido su avance en diferentes puntos.

Tales inquietudes son comunes con cualquier cliente y en cualquier proyecto de desarrollo que requiera una considerable cantidad de tiempo, dinero y recursos humanos.

# Metodologías: antiguas y recientes

A usted no le gustaría que el equipo de desarrollo comenzara a codificar sin más. Después de todo, ¿qué sería lo que codificarían? El equipo de desarrollo tiene que proceder de manera más estructurada y metódica. La estructura y naturaleza de los pasos en un esfuerzo de desarrollo es lo que yo entiendo como *metodología*.

Antes de comenzar a programar, los desarrolladores tienen que comprender con claridad el problema. Esto requiere que alguien analice sus requerimientos. Una vez hecho ese análisis, ¿se podrá iniciar la codificación? No. Alguien tiene que convertir tal análisis a diseño. De esta manera, los codificadores comenzarán a producir el código a partir del diseño, después de probar y distribuir el código se convertirá en un sistema.

## El método antiguo

Esta idea demasiado simplificada del proceso de desarrollo podrá darle una idea de que las etapas deberán sucederse en lapsos claramente definidos, una después de otra. De hecho, las metodologías de desarrollo iniciales se estructuraban de esa manera. La figura 15.1 le muestra una forma de pensar cuya influencia trascendió por varios años. Éste es el método "en cascada", y establece que el análisis, diseño, codificación y distribución van uno después de otro como las actividades en un diagrama de actividades: solamente cuando se haya completado uno se podrá iniciar el otro.

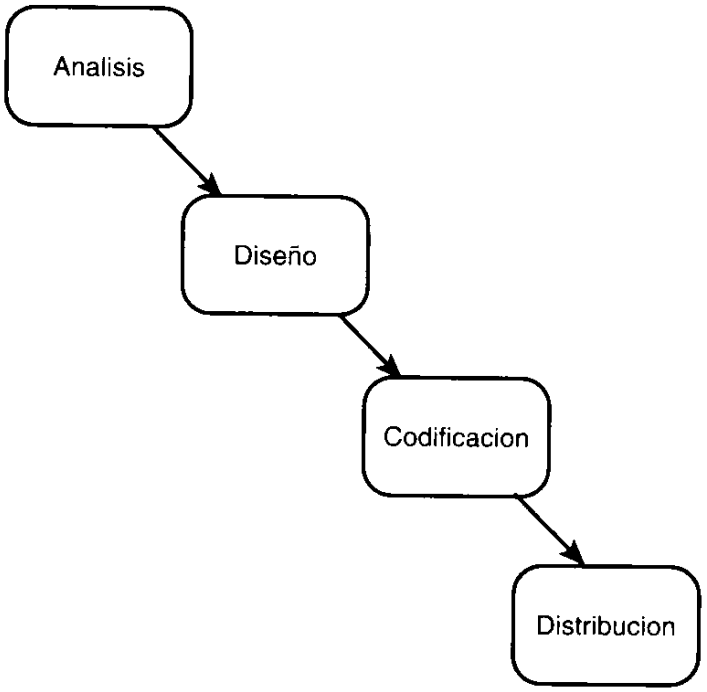
Esta forma de hacer las cosas tiene algunos puntos inquietantes. Por un lado, tiende a la realización de tareas individuales. Si un analista no tiene contacto con un diseñador, y éste a su vez no tiene contacto con un desarrollador, existe la posibilidad de que los tres miembros rara vez trabajen juntos para compartir puntos de vista importantes.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 15.1**

*El método "en cascada" del desarrollo del software.*




Otro problema con este método es que reduce el impacto de la comprensión obtenida en el proyecto. (No se equivoque, la comprensión evoluciona durante la vida de un proyecto —aun después de que un análisis se haya volcado en un diseño—). Si el proceso no puede retroceder y volver a ver los primeros estados, es posible que las ideas desarrolladas no sean utilizadas. Intentar introducir con calzador nuevos puntos de vista durante el desarrollo es, cuando menos, bastante difícil. La revisión de un análisis y diseño —y la ulterior incorporación de una idea desarrollada— establece una mayor oportunidad de éxito.

## El método reciente

En contraste con el método de cascada, la moderna ingeniería de programas tiende a la colaboración entre las fases del desarrollo. Los analistas y los diseñadores, por dar un ejemplo, hacen revisiones para desarrollar un sólido fundamento para los desarrolladores. Éstos, a su vez, interactúan con los analistas y los diseñadores para compartirles sus puntos de vista, modificar los diseños y fortalecer su código.

La ventaja es que conforme crece la comprensión, el equipo incorpora nuevas ideas y genera un sistema más confiable. La desventaja (en caso de haberla) es que algunas personas no son muy participativas y pueden mantenerse al margen. En ocasiones, los gerentes de proyectos quisieran decirle a sus clientes: "Ya se finalizó el análisis y ahora continuaremos con el diseño. Luego de unos dos o tres días de diseño, empezaremos a codificar."

Tal mentalidad es peligrosa. Establecer barreras artificiales entre las fases podría dar por resultado un sistema que no haga exactamente lo que los clientes desean.

www.elsolucionario.net

www.elsolucionario.net

El método antiguo fomenta otro problema: es común el caso de que los partidarios al método "en cascada" reparten el tiempo del proyecto en la codificación. El verdadero efecto de esto es que se quita un tiempo valioso al análisis y diseño.

# Lo que debe hacer un proceso de desarrollo

En los primeros años de la programación de computadoras, una persona podía analizar un problema, otorgar una solución y escribir un programa. En los primeros años de la construcción de viviendas (cuando la tierra era plana), una persona podía construir una casa, también.

En la actualidad la historia ha cambiado. Para desarrollar las diferentes naturalezas de sistemas complejos que demandan los negocios de hoy, es más necesario el uso de un equipo. ¿Por qué? El conocimiento se ha especializado tanto que una persona no puede conocer todas las facetas de un negocio, comprender un problema, diseñar una solución, traducirla a un programa, distribuir el programa en el hardware y asegurarse que los componentes del hardware funcionen de forma conjunta.

El equipo tiene que formarse de analistas para comunicarse con el cliente y comprender el problema, diseñadores para generar una solución, programadores para codificarla e ingenieros de sistemas para distribuirla. Un proceso de desarrollo tiene que tomar en cuenta todos los procesos anteriores, utilizarlos adecuadamente y asignar la cantidad de tiempo necesaria para cada fase. El proceso también debe dar por resultado diversos productos del trabajo que den indicios de progreso y conformar una estela de responsabilidad.

Finalmente, el proceso deberá asegurar que sus fases no sean discontinuas. En lugar de ello, debe obtenerse información entre las fases para fomentar la creatividad y aumentar la facilidad de innovar. La base sería: es más sencillo hacer una modificación a un proyecto y luego hacerla en la casa, en lugar de modificar la casa mientras construye la estructura.

Al llegar a un proceso, existe la tentación de generar una serie de fases que podrían traer una gran cantidad de papeleo. Algunas metodologías que están disponibles de forma comercial lo hacen, con lo que hacen que los gerentes de proyectos llenen interminables formularios. El papeleo se complica por sí mismo.

Una razón de esto es la idea errónea de que la metodología de la "unitalla" es posible; cada empresa es única. Una empresa tiene su propia cultura, normatividad, historia y personal. La metodología del desarrollo que pueda aplicarse a un consorcio internacional posiblemente fallará en una pequeña empresa y viceversa. Al intentar meter con calzador una metodología en una empresa, se tendrá la mala impresión de que un papeleo extremo podrá ayudar.

www.elsolucionario.net

www.elsolucionario.net

Así que aquí está el reto. Un equipo de desarrollo deberá:

- Asegurar que el equipo de desarrollo cuenta con una firme comprensión del problema que se intenta resolver
- Dar pie a un equipo que conste de una colección de responsabilidades
- Fomentar la comunicación entre los miembros del equipo que ostentan tales responsabilidades
- Dar pie a la intercomunicación entre las fases del proceso de desarrollo
- Desarrollar productos de trabajo que comuniquen el progreso al cliente, y eliminar el papeleo superfluo

¡Ah! Por cierto, sería una buena idea si el proceso origina un producto terminado en un lapso corto.

Se habrá dado cuenta que utilizo las palabras *proceso* y *metodología* de forma indistinta. Aunque es posible encontrar algunas diferencias entre ellas, prefiero no discutir los detalles. En mi experiencia, la palabra *metodología* se ha tergiversado de forma paulatina. Creo que si se mezcla la palabra *proceso* en el tema, se puede eliminar tal tergiversación.

# GRAPPLE

Para enfrentar este reto de varias facetas, le presento GRAPPLE (Guías para la Ingeniería de Aplicaciones Rápidas)). Las ideas dentro de GRAPPLE no son originales. Son una condensación de las ideas de varias otras personas. Los Tres Amigos han creado el Proceso Racional Unificado, y antes de ello, cada Amigo tenía su propio proceso; las ideas en tales procesos son similares a GRAPPLE. El libro de Steve McCornell, *Rapid Development* (Microsoft Press, 1996), contiene varias de las mejores prácticas que se aplican al... bueno... desarrollo rápido de aplicaciones.

La primera palabra en las siglas de GRAPPLE, *Directivas*, es importante: ésta no es una férrea metodología. En vez de ello, es un conjunto de ideas adaptables y flexibles. Imagínelas como un patrón simplificado de un proceso de desarrollo. Lo presento como un vehículo para mostrar al UML dentro de un contexto. Con algunos ajustes, GRAPPLE puede aplicarse en diversas organizaciones (aunque, tal vez, no a todas). Da la oportunidad a un gerente de proyectos, con creatividad, de agregar sus propias ideas respecto a lo que funcionará en una organización en particular, y puede sustraer los pasos incluidos que no funcionen.

www.elsolucionario.net

www.elsolucionario.net

Antes de adentrarnos en el tema de GRAPPLE, he aquí una pregunta que tal vez se formule: "¿Por qué me está hablando de esto un libro que, se supone, trata del UML?"

La respuesta es que si no le digo a usted algo respecto al proceso de desarrollo y le doy un contexto para utilizar al UML, todo lo que habré hecho es mostrarle cómo dibujar diagramas. Lo importante de esto es mostrarle cómo y cuándo utilizar cada uno de ellos.

En la parte II, "Estudio de un caso", verá un caso de prueba que aplicará a GRAPPLE en el UML.

## RAD³: la estructura de GRAPPLE

GRAPPLE consta de cinco *segmentos*. He utilizado "segmentos" en lugar de "fases" para eliminar la idea de que una "fase" debe completarse antes de iniciar la otra. (Resistí la tentación de llamarlos "piezas". "Cinco piezas fáciles " era demasiado hermoso.) Cada segmento, en turno, consta de diversas *acciones*. Cada acción trae consigo un *producto del trabajo*, y cada acción es responsabilidad de un *jugador*.

En muchos casos, el gerente de proyectos puede combinar los productos de trabajo en un informe que presente al cliente. Los productos de trabajo, de hecho, tienen el mismo propósito que un avance en papel, sin sumergirse en el papeleo.

Para adaptar a GRAPPLE, un gerente de proyectos podría agregar acciones a cada segmento. Hay otra posibilidad, que es profundizar a un nivel inferior, y subdividir a cada acción en subacciones. Aun hay otra posibilidad de reordenar las acciones dentro de cada segmento. Las necesidades de organización establecerán el camino a seguir.

GRAPPLE se encausa a los sistemas orientados a objetos. Por ello, las acciones dentro de cada segmento se orientan a crear productos de trabajo de una naturaleza orientada a objetos.

Los segmentos son:

1. Recopilación de necesidades
2. Análisis
3. Diseño
4. Desarrollo
5. Distribución

www.elsolucionario.net

www.elsolucionario.net

Esto nos otorga un acrónimo RADDD o RAD³. Luego del tercer segmento, el gerente de proyectos combina los productos de trabajo en un documento de diseño para dárselo al cliente y los desarrolladores. Cuando se han completado todos los segmentos RAD³, todos los productos de trabajo se combinan para conformar un documento que define al sistema.

Antes de iniciar tales segmentos, debe asumir que el cliente ha generado un caso de negocios para el nuevo sistema. También debe asumir que los miembros del equipo de desarrollo, particularmente los analistas, han leído tanta documentación relevante como sea posible.

Examinemos cada segmento con mayor atención, sin perder de vista las partes del UML que se ajusten a cada uno.

# Recopilación de necesidades

Si intenta asignar una importancia relativa a cada segmento, éste es un buen candidato para ser el *número uno*. Si no comprende lo que desea el cliente, nunca podrá generar el sistema adecuado. Todos los análisis de casos de uso en el mundo no le ayudarán si no comprende las bases del dominio del cliente y el problema que quiera que usted resuelva.

## Descubra los procesos de negocios

Es una buena idea empezar el proceso de desarrollo mediante la comprensión de los procesos de negocios del cliente, en especial aquellos que tratará de mejorar con el sistema propuesto. Para comprenderlo, un analista entrevistará al cliente o a una persona con el conocimiento necesario que sea designada por el cliente, a quien le preguntará los pasos relevantes del proceso uno por uno.

Una consecuencia importante será que el analista obtendrá un vocabulario de trabajo en un subconjunto de la terminología del cliente. El analista usará este vocabulario cuando entreviste al cliente en la siguiente acción.

El producto del trabajo es un diagrama de actividades o conjunto de ellos que captan los pasos y puntos decisivos en el proceso (o procesos) del negocio.

## Realice un análisis del dominio

Esta acción es como el ejemplo de la plática con el entrenador de baloncesto. Puede realizarse durante la misma sesión en la acción anterior. El objetivo es comprender de la mejor manera posible el dominio del cliente. Observe que esta acción y la anterior tratan de conceptos, no del sistema que va a generar. El analista tiene que acomodarse en el mundo del cliente, pues, a fin de cuentas, es el interlocutor entre el cliente y el equipo de desarrollo.

www.elsolucionario.net

www.elsolucionario.net

El analista entrevista al cliente con la finalidad de comprender las principales entidades en el dominio del cliente. Durante la plática entre el cliente y el analista, otro miembro del equipo tomará las notas (de forma óptima) en un equipo de cómputo portátil equipado con un procesador de textos, y un modelador de objetos generará un diagrama de clases de alto nivel. Si puede contar con más de un miembro del equipo que tome notas, por su bien hágalo.

El modelador de objetos prestará atención a los sustantivos y empezará a convertir a cada uno en una clase. Finalmente, algunos de esos sustantivos se convertirán en atributos. El modelador también prestará atención a los verbos, que se convertirán en operaciones de las clases. En este punto, una herramienta de modelado computarizada sería muy útil.

El producto del trabajo es un diagrama de clases de alto nivel y un conjunto de minutas.

**¿GRABAR O NO GRABAR?**

**¿Debería grabar en cinta de sonido tales entrevistas o tan sólo basarse en las minutas? Ésta es una pregunta común. Cuando se graba una entrevista, se tiende a no prestar tanta atención, o a sólo tomar algunas notas muy obligatorias. (Después de todo, siempre podrá recurrir a la cinta). Si decide grabar, le sugiero que ignore a la grabadora y tome todas las notas como si la grabadora no existiera.**

**La grabación en cinta de sonido puede ser muy útil cuando se encuentra en proceso de entrenamiento de un nuevo modelador de objetos. Un modelador experimentado podrá comparar los diagramas del nuevo mediante la grabación del debate y verificar su minuciosidad.**

## Identificación de los sistemas cooperativos

El poeta del s. XVII John Donne, escribió: "Nadie es una isla, completo en sí mismo". Si él hubiera escrito en la época actual, tal texto debió decir "Ninguna *persona* es una masa de tierra rodeada de agua, completa en sí misma". También pudo haber escrito "Ningún *sistema* es una isla...", y así por el estilo.

En cualquier caso Donne estaría en lo correcto. Normalmente, los sistemas de negocios actuales no emergen de la nada, tienen que colaborar con otros. En las primeras instancias del proceso, el equipo de desarrollo verá exactamente de qué sistemas dependerá el nuevo sistema, y cuáles dependerán de él. Un diseñador de sistemas se encargará de esto, y producirá un diagrama de distribución como su producto del trabajo. El diagrama muestra a los sistemas como nodos, con líneas de comunicación entre ellos, componentes residentes y dependencias entre componentes.

www.elsolucionario.net

www.elsolucionario.net

## Descubra las necesidades del sistema

Descubrir las necesidades es muy importante, ya que en esta acción, el equipo realiza su primera sesión de JAD (Desarrollo conjunto de aplicaciones). Habrá otras más en el curso del GRAPPLE.

Una sesión JAD reúne a quienes toman las decisiones en la empresa del cliente, a los usuarios potenciales y a los miembros del equipo de desarrollo. Debe haber alguien que modere la sesión; el trabajo del moderador es obtener una respuesta de quienes toman las decisiones y de los usuarios acerca de lo que esperan que haga el sistema. Al menos deberá haber dos miembros del equipo que tomen notas, y el modelador de objetos deberá refinar el diagrama de clases que se obtuvo previamente.

El producto del trabajo es un diagrama de paquetes. Cada paquete representa a un área de alto nivel de la funcionalidad del sistema (por ejemplo: "ayuda con el servicio a clientes"). Cada paquete agrupa un conjunto de casos de uso (por ejemplo: "obtener el historial del cliente" o "tratar con el cliente").

La complejidad del sistema será lo que determine la duración de la sesión. Casi nunca será menor a medio día laboral, y podría durar hasta toda una semana laboral. La empresa del cliente debe hacer el compromiso de invertir el tiempo que sea necesario.

¿Para qué acceder a una sesión JAD para desarrollar los requerimientos del sistema? ¿Por qué no sólo entrevistar a cada individuo? Como podrá recordar, dije que la última parte del reto de un proceso de desarrollo es generar un sistema en un corto lapso. Las entrevistas individuales pueden tardar semanas, mucho más si existen conflictos en los itinerarios de las personas. La espera de entrevistas individuales puede ocupar mucho tiempo y, con él, puede irse por tierra la supuesta ventaja competitiva de completar rápidamente el sistema. Las entrevistas individuales posiblemente contendrían puntos de vista conflictivos, y se perdería tiempo en intentar resolverlos. Agruparlos a todos crea una expectativa general, en la que los participantes podrían hacer una simbiosis de sus puntos de vista en beneficio de todos.

www.elsolucionario.net

## Presentar los resultados al cliente

Cuando el equipo finaliza todas las acciones de Necesidades, el administrador de proyectos presentará los resultados al cliente. Algunas empresas podrían requerir la aprobación del cliente en este punto, para que pueda proceder el desarrollo. Otras podrían necesitar una estimación de los costos de acuerdo con los resultados. De esta manera, el producto del trabajo podría variar de acuerdo con la empresa.

# Análisis

En este segmento, el equipo profundiza en los resultados del segmento Necesidades y aumentará su comprensión del problema. De hecho, partes de este segmento empezarán durante el segmento de Necesidades, conforme el modelador de objetos empieza a depurar el diagrama de clases durante la sesión JAD de Necesidades.

www.elsolucionario.net

## Comprensión del uso del sistema

Esta acción es un análisis de casos de uso de alto nivel. En una sesión JAD con usuarios potenciales, el equipo de desarrollo trabaja con los usuarios para descubrir a los actores que iniciarán cada caso de uso, y los actores que serán beneficiados. (Recuerde que un actor puede ser un sistema o una persona.) Un moderador interviene en la sesión, y dos miembros del equipo toman notas. Luego de algunos proyectos, el moderador de esta sesión podría convertirse en el analista de casos de uso.

El equipo también intentará desarrollar nuevos casos de uso y casos de uso abstractos. El producto del trabajo será un conjunto de diagramas de casos de uso que muestren a los actores y las dependencias estereotipadas ("extender" e "incluir") entre los casos de uso.

## Hacer realidad los casos de uso

En esta acción, el equipo de desarrollo continúa su trabajo con los usuarios. El objetivo es analizar la secuencia de pasos en cada caso de uso. Esta sesión JAD puede ser la continuación de la sesión previa. Cuidado: ésta es, por lo general, la sesión JAD más compleja para los usuarios. Tal vez no estén acostumbrados a dividir una operación en los pasos que la conforman y, a su vez, tampoco puedan enumerarlos. El producto del trabajo es una descripción textual de los pasos en cada caso de uso.

## Depurar los diagramas de clases

Durante las sesiones JAD, el modelador de objeto escuchará todos los debates y continuará con su depuración del diagrama de clases. En este punto, el modelador de objetos deberá rellenar los nombres de las asociaciones, clases abstractas, multiplicidades, generalizaciones y agregaciones. El producto del trabajo es un diagrama de clases depurado.

## Analizar cambios de estado en los objetos

El modelador de objetos depurará el modelo mediante la presentación de cambios de estado conforme sea necesario. El producto del trabajo es un diagrama de estados.

## Definir la comunicación entre objetos

Ahora que el equipo cuenta con un conjunto de diagramas de casos de uso y un diagrama depurado de clases, se definirá la forma en que los objetos se comunican. El modelador de objetos desarrollará un conjunto de diagramas de secuencias y de colaboraciones para delinear la comunicación. Deberán incluirse los cambios de estado. Estos diagramas conforman el producto del trabajo de esta acción.

www.elsolucionario.net

www.elsolucionario.net

### Analizar la integración con diagramas de colaboraciones

Al tiempo de realizar los pasos anteriores, el diseñador del sistema descubre los detalles específicos de la integración con los sistemas cooperativos. ¿Qué tipo de comunicación está envuelto? ¿Cuál es la arquitectura de red? Si el sistema tendrá que utilizar bases de datos, un analista de bases de datos determinará la arquitectura (física o lógica) de ellas. Los productos del trabajo son diagramas de distribución detallados y (de ser necesario) modelos de datos.

## Diseño

En este segmento, el equipo trabajará con los resultados del segmento de Análisis para diseñar la solución. En el diseño y en el análisis se harán las revisiones pertinentes hasta que el diseño se haya completado. De hecho, algunas de las metodologías combinan al Análisis y al Diseño en una sola fase.

### Desarrollo y depuración de los diagramas de objetos

Los programadores tomarán el diagrama de clases y generarán cualesquier diagramas de objetos que sea necesario. Darán vida a los diagramas de objetos mediante el análisis de cada operación y el desarrollo de un diagrama de actividades correspondiente. Los diagramas de actividades fungirán como la base de gran parte del código en el segmento de desarrollo. Los productos del trabajo serán los diagramas de objetos y los de actividades.

### Desarrollo de diagramas de componentes

Los programadores serán quienes jueguen un importante papel en esta acción. La tarea será visualizar los componentes que resultarán del siguiente segmento y mostrar las dependencias entre ellos. Los diagramas de componentes serán el producto del trabajo.

### Planeación para la distribución

Cuando se haya completado el diagrama de componentes, el diseñador del sistema empezará a planear la distribución e integración con sistemas cooperativos. Creará un diagrama de distribución que muestre el lugar donde se encontrarán los componentes. El producto del trabajo será un diagrama que sea parte del de distribución generado con anterioridad.

### Diseño y prototipos de la interfaz del usuario

Esto trae consigo otra sesión JAD con los usuarios. Aunque esto es parte del Diseño, esta sesión puede ser la continuación de anteriores sesiones JAD con los usuarios —un indicio de la interacción entre el Análisis y el Diseño—.

www.elsolucionario.net

www.elsolucionario.net

La interfaz del usuario debería permitir la consumación de todos los casos de uso. Para ello, un analista de GUI deberá trabajar con los usuarios para desarrollar prototipos, en papel, de las pantallas que corresponderán a grupos de casos de uso. Los usuarios pegarán papeletas removibles que representen los componentes de la pantalla (botones, casillas de verificación, listas desplegables, menús y cosas así). Cuando los usuarios queden satisfechos de la posición de los componentes, los desarrolladores generarán prototipos de las pantallas para que sean aprobados por los usuarios. Los productos del trabajo serán capturas de pantalla de los prototipos resultantes.

### Pruebas de diseño

Los casos de uso permiten el diseño de pruebas del software. El objetivo es evaluar si el software hace lo que se supone que debería (esto es, que hace lo que se especifica en los casos de uso). Preferentemente, un desarrollador o especialista de pruebas externo al equipo de desarrollo deberá utilizar los diagramas de casos de uso para crear secuencias de comandos en herramientas automatizadas de pruebas. Tales secuencias de comandos conformarán el producto del trabajo.

### Iniciar la documentación

Nunca es demasiado pronto para empezar a documentar el sistema para los usuarios finales y gerentes de sistemas. Los especialistas en la documentación trabajarán en conjunto con los diseñadores para empezar a generar un panfleto de la documentación y llegar a una estructura de alto nivel para cada documento. Tal estructura es el producto del trabajo.

## Desarrollo

De este segmento se encargan los programadores. Con suficiente análisis y diseño, este segmento debería realizarse con rapidez y sin problemas.

### Generación del código

Con los diagramas de clases, de objetos, de actividades y de componentes a la mano, los programadores generarán el código del sistema. Tal código es el producto del trabajo de esta acción.

### Verificación del código

Los especialistas en pruebas (no los desarrolladores) ejecutarán secuencias de comandos de prueba para evaluar si el código hace lo que se pretende. Los resultados de las pruebas son los productos del trabajo. Esta acción alimenta a la anterior y viceversa, hasta que el código pase todos los niveles de prueba.

### Generación de interfaces del usuario, conexión con el código y prueba

Esta acción crea las interfaces de usuario ya aprobadas. El especialista en GUI las genera y conecta con el código. Las pruebas ulteriores aseguran que las interfaces funcionen adecuadamente. El sistema en funcionamiento junto con las interfaces de usuario, son el producto del trabajo.

www.elsolucionario.net

www.elsolucionario.net

### Consumación de la documentación

Durante el segmento de desarrollo, los especialistas en documentación trabajan en paralelo con los desarrolladores para asegurar la entrega oportuna de toda la documentación, la cual es el producto del trabajo de esta acción.

## Distribución

Cuando un sistema se ha finalizado, se distribuye en el hardware adecuado y se integra con los sistemas cooperativos. No obstante, la primera acción en este segmento puede iniciar antes de que el segmento de Desarrollo comience.

### Planeación para copias de seguridad y recuperación

El diseñador del sistema creará un plan que incluya los pasos a seguir en caso de que el sistema falle. El plan, producto del trabajo de esta acción, establece lo que se deberá hacer para crear una copia de seguridad del sistema y para recuperarse del error.

### Instalación del sistema terminado en el hardware adecuado

El diseñador del sistema, con toda la ayuda necesaria de los programadores, distribuye el sistema terminado en los equipos de cómputo adecuados. El producto del trabajo es el sistema completamente distribuido.

### Verificación del sistema instalado

Finalmente, el equipo de desarrollo verifica el sistema instalado. ¿Se ejecuta como se esperaba? ¿El plan de copias de seguridad y recuperación funciona? Los resultados de estas pruebas determinarán si se necesita hacer una depuración ulterior. Tales resultados conforman el producto del trabajo de esta acción.

### Celebración

Sin mayor explicación, el equipo de desarrollo podrá inventar los productos del trabajo de esta acción.

www.elsolucionario.net

# Resumen de GRAPPLE

Si revisa los segmentos y acciones en GRAPPLE, verá que los movimientos van de lo general a lo específico: de lo rústico a lo refinado. Empieza con una asimilación conceptual del dominio, trasciende a la funcionalidad de alto nivel, profundiza en los casos de uso, depura los modelos y diseña, desarrolla y distribuye el sistema.

También notó que hay más acciones en los segmentos de Análisis y Diseño que en el de Desarrollo. Esto es por diseño, valga la redundancia. La idea es utilizar tanto tiempo como sea necesario en el análisis y diseño, para que la codificación se realice sin problemas. Podría parecer una herejía, pero en el mundo ideal la codificación es sólo una pequeña parte del desarrollo de sistemas. Entre más analice, más cerca estará del ideal.

www.elsolucionario.net

GRAPPLE, como lo dije, es un patrón simplificado de un proceso de desarrollo. No me centré en los detalles en ciertos puntos importantes, como los niveles de prueba. También pasé por alto algunas cuestiones básicas: ¿Dónde y cómo el equipo alberga los producto del trabajo en ejecución? ¿Cómo trata el equipo el importante punto de la administración de la configuración?

No aludí a estos puntos porque se salen del tema del UML. Una respuesta corta para estos puntos importantes es cobijarse en la tecnología. Los productos del trabajo (finalizados o en ejecución) pueden encontrarse en un área de almacenamiento que se encuentre en la red de la empresa. Una opción es contar con una jerarquía de directorios a la que puedan acceder los miembros del equipo. Una opción más segura es instalar un paquete de almacenamiento central que lleve un control del cumplimiento e inicio de los productos del trabajo, y que sólo permita que una persona a la vez verifique una copia modificable de un elemento. Éste es el fundamento de una solución para la administración de la configuración. La tecnología de almacenamiento avanza con regularidad, y hay varias opciones.

La hora siguiente dará inicio a la parte II, el estudio de un caso que aplica tanto al UML como a GRAPPLE.

# Resumen

Una metodología de desarrollo estructura los segmentos y actividades en un proyecto de desarrollo de sistemas. Sin una metodología habría un caos y los desarrolladores no comprenderían el problema que se supone deberían resolver, así como los sistemas no cumplirían con las necesidades de los usuarios. Las metodologías de antaño forzaban a una secuencia "en cascada" de análisis, diseño, codificación y distribución.

Este tipo de metodología secuencial podía fragmentar el desarrollo, de modo que un equipo de desarrollo podría no aprovechar la mejor asimilación que se obtiene durante la vida de un proyecto. Por lo general, también distribuye la mayor parte del tiempo en la codificación, y esto resta una enorme cantidad de tiempo al análisis y diseño.

Esta hora presentó al GRAPPLE (Directivas para el Rápido Diseño de Aplicaciones), un patrón para el proceso de desarrollo. GRAPPLE consta de cinco segmentos: Recopilación de necesidades, Análisis, Diseño, Desarrollo y Distribución. Cada segmento consta de diversas acciones, y cada una de ellas da por resultado un producto del trabajo. Los diagramas UML constituyen productos del trabajo para varias de las acciones.

La parte II aplica a GRAPPLE y el UML al estudio de un caso.

www.elsolucionario.net

www.elsolucionario.net

# Preguntas y respuestas

**P ¿En algún momento se podría aplicar el método "en cascada"?**

**R** Si el ámbito del sistema propuesto es muy pequeño (claro que es algo subjetivo), podría aplicarlo sin problemas. No obstante, en el moderno desarrollo de sistemas orientados a objetos, una metodología que propenda a la interacción entre los segmentos de desarrollo podrá otorgar un mejor resultado.

**P En la respuesta anterior, mencionó el desarrollo de sistemas orientados a objetos. Suponga que el sistema propuesto no está orientado a objetos. ¿Aún así se aplica?**

**R** Incluso en los sistemas que no estén orientados a objetos (como en muchos proyectos basados en grandes computadoras centralizadas) se aplican las ideas que ha visto en esta hora. Las sesiones JAD, un análisis y diseño de avanzada y la interacción entre los segmentos de desarrollo inclusive serán muy útiles. Podría adaptar a GRAPPLE (por ejemplo: mediante la eliminación del modelado de clases), pero esa es la idea: es un conjunto de directivas flexibles en lugar de ser una metodología que tenga que seguirse a pie juntillas.

# Taller

Ahora que ya comprendió las metodologías, verifique, con las siguientes preguntas, qué tanto ha asimilado. El apéndice A, "Respuestas a los cuestionarios", le dará las respuestas.

## Cuestionario

1. ¿Cuáles son algunas de las inquietudes de un cliente?
2. ¿Qué se debe comprender como *metodología* de desarrollo?
3. ¿Cuál es el método "en cascada"? ¿Cuáles son sus debilidades?
4. ¿Cuáles son los segmentos de GRAPPLE?
5. ¿Qué es una sesión JAD?

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# PARTE II
# Estudio de un caso

## Hora


16 Presentación del caso por estudiar

17 Elaboración de un análisis de dominio

18 Recopilación de las necesidades del sistema

19 Desarrollo de los casos de uso

20 Orientación a las interacciones y cambios de estado

21 Diseño del aspecto, sensación y distribución

22 Noción de los patrones de diseño


www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 16

# Presentación del caso por estudiar

Ahora que ya cuenta con cierta experiencia del UML y se le ha presentado el patrón de una metodología de desarrollo, verá cómo aplicar el UML en un proceso de desarrollo. A partir de aquí daremos inicio a la parte II, el estudio de un caso que aplica al UML bajo el contexto del proceso GRAPPLE.

En esta hora se tratarán los siguientes temas:

- El panorama del caso por estudiar
- Cómo descubrir y modelar los procesos del negocio
- Sugerencias al hacer entrevistas

La empresa multinacional (y ficticia) LaHudra, Nar y Goniff, S.A., ha hecho una encuesta sobre el mundo de los restaurantes y ha llegado a sorprendentes conclusiones: a la gente le gusta comer fuera, pero no disfrutan algunos momentos de esa experiencia.

"Bueno", dijo LaHudra, "pude haber predicho los resultados de nuestra encuesta. Cuando salgo a comer, me disgusta cuando el mesero toma mi orden y se desaparece por una hora. Al ir uno a un restaurante con clase se espera un trato diferente".

www.elsolucionario.net

www.elsolucionario.net

"Cierto", respondió Nar, "En ocasiones cambio de parecer luego de hacer mi orden y quiero decirle al mesero que aguarde. O tengo una pregunta... o algo... y no puedo encontrarlo."

Goniff asiente: "Estoy de acuerdo. Pero, con todo, el comer fuera es divertido; me agrada cuando alguien me sirve y además, me gusta la idea de que el personal de cocina me prepare la comida. Los resultados de nuestra encuesta muestran que la mayoría de las personas también opinan de esa forma."

"¿No habría alguna manera en que podamos mantener la experiencia básica y mejorarla de alguna forma?"

"¿Cómo?", preguntó Nar.

"¡Yo sé cómo!", dijo LaHudra. "Con tecnología."

Y así fue que decidieron que uno de sus equipos de desarrollo de software corporativo construyera el restaurante del futuro.

# Aplicación de GRAPPLE al problema

Los miembros del equipo de desarrollo se ciñen al esquema de GRAPPLE. Saben que la mayor parte del tiempo en el proyecto deberán orientarlo al análisis y diseño. De esa forma, la codificación se generará con rapidez y eficiencia, con lo que aumentará la posibilidad de una instalación y distribución sin problemas.

El proyecto debe iniciar con la recopilación de necesidades, y con la asimilación del dominio del restaurante. Como podrá recordar de la hora anterior, la recopilación de necesidades consta de las siguientes acciones:

- Descubrir los procesos del negocio
- Realizar un análisis del dominio
- Identificar los sistemas cooperativos
- Descubrir los requerimientos del sistema
- Presentar los resultados al cliente

En esta hora trataremos la primera acción.

# Descubrir los procesos del negocio

LaHudra, Nar y Goniff lo hacen todo a lo grande. Están listos para entrar en el mundo de los restaurantes y han conformado una División de Restaurantes LNG. Han contratado a varios restauranteros, meseros (camareros), chefs y personal de mantenimiento experimentados.

Todo lo que esperan es el apoyo tecnológico para el restaurante del futuro. Luego, establecerán su primer restaurante, íntegro, con todo y la tecnología para mejorar el placer de comer en un restaurante.

www.elsolucionario.net

www.elsolucionario.net

Los miembros del equipo de desarrollo están de suerte. Iniciarán con un papel en blanco. Todo lo que deberán hacer será comprender los procesos del negocio y el dominio para continuar en esa línea.

El análisis del proceso del negocio empieza con la entrevista de un analista a un restaurantero. Durante la entrevista, alguien tomará nota en una computadora portátil. Al mismo tiempo, un modelador plasmará en una pizarra blanca un diagrama de actividades que el analista, quien está tomando nota y el restaurantero podrán ver.

En las siguientes subsecciones, seguiremos la entrevista en cada proceso del negocio en un restaurante. La meta es producir los diagramas de actividades que modelen los procesos.

## Servir a un cliente

"Gracias por atendernos", dice el analista.

"Es un placer", dice el restaurantero. "¿Qué es exactamente lo que desean saber?"

"Empecemos con una sencilla transacción de negocios. ¿Qué sucede cuando un cliente entra al restaurante?"

"Bueno, si el cliente tiene un abrigo o chaqueta, le ayudamos a quitárselo, lo almacenamos en un guardarropa y le damos un boleto para solicitarlo posteriormente. Eso mismo hacemos con un sombrero. Luego, nosotros..."

"Un momento. Suponga que hay una línea de espera. ¿Primero se forma, o da su nombre al capitán, o...?"

"No. Intentamos que se sienta tan cómodo como sea posible al llegar. Luego nos preocupamos por la línea de espera, en caso de haber alguna"

"De hecho, si hay una lista de espera, le preguntamos al cliente si hizo alguna reservación. Si la hizo, intentamos honrarlos de forma oportuna y darles asiento tan pronto como sea posible. Si no hay una reservación, deja su nombre y puede ir a nuestro bar para tomar algo antes de comer. Claro que no es obligatorio que lo hagan. Pueden tan sólo sentarse en un área de espera claramente indicada."

"Interesante. Aún no se han sentado a comer y ya se han logrado algunos puntos decisivos."

Hagamos una pausa para determinar a dónde llegamos. El diagrama del proceso del negocio ahora luce como en la figura 16.1.

De vuelta a la entrevista.

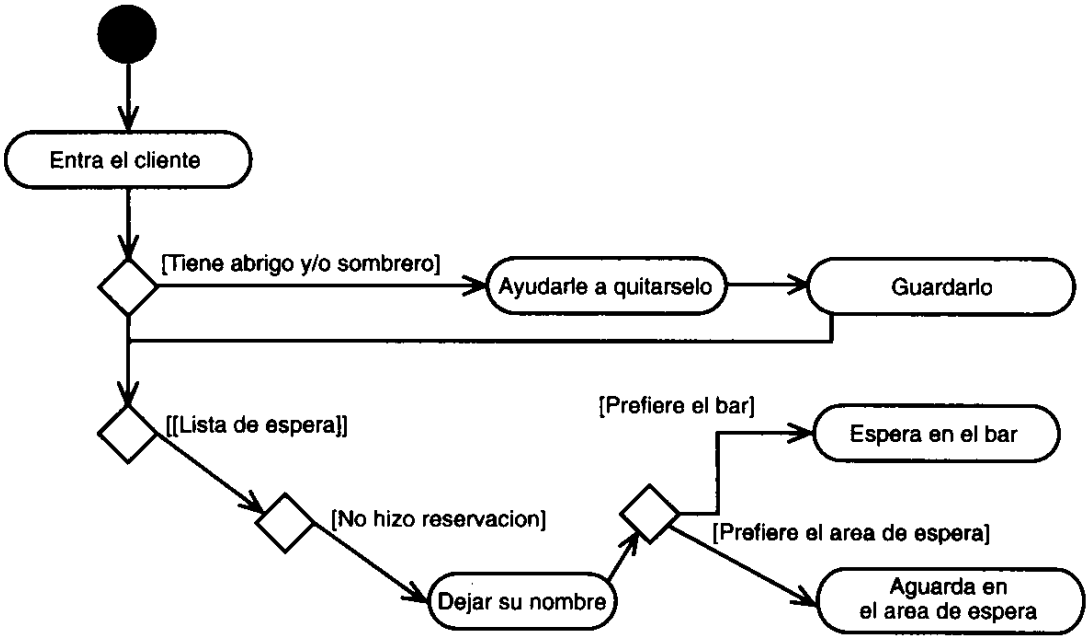
El trabajo del analista es continuar por el proceso del negocio.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 16.1**

*Las fases iniciales del diagrama de actividades del proceso del negocio del restaurante, "Servir a un cliente".*




"Bien. Cuando le llegue el turno al cliente, o que haya llegado un cliente que hizo reservación, será hora de sentarlo, ¿no?"

"Sí, pero ahora que lo pienso no es tan sencillo. La mesa deberá estar lista; para ello, deberá ser limpiada. Un mozo de piso debe cambiar el mantel usado por el cliente anterior, y acomodar la mesa. Cuando está lista, el capitán de meseros lleva al cliente a su mesa y luego llama a un mesero." "¿lo llama?"

Observe lo que hace el analista. El restaurantero ha utilizado un nuevo término (nuevo dentro del contexto de la entrevista), y el analista desea que se lo defina.

El saber cuándo y cómo hacer esto es parte del arte de entrevistar, donde la experiencia será el mejor maestro.

"Sí. No es muy complicado dado que los meseros tienen sus áreas asignadas de servicio y, por lo general, saben cuándo está lista una mesa. Normalmente circundan su área y están atentos a las expresiones del capitán."

"¿Luego qué ocurre?"

"Bueno, el mesero llega a la mesa, entrega un menú a cada comensal y les pregunta si desean alguna bebida mientras se deciden. Luego llamará a un 'asistente', quien colocará una charola con pan y mantequilla, y llenará un vaso con agua para cada persona en la mesa. Si alguien ha pedido una bebida, el mesero la traerá."

www.elsolucionario.net

www.elsolucionario.net

"Un momento. Dijo 'el mesero'. ¿La persona que sirve siempre es un hombre?"

"No. Hablo de 'el mesero' por hábito. Lo siento."

"Bien. ¿Qué hay si utilizamos el término neutral 'sirviente'? También vi que el cliente tiene un par de oportunidades para pedir una bebida."

"Sirviente no es una palabra que usemos en el ámbito restaurantero. Sugiero que sigamos con 'mesero' y que se aplique indistintamente a hombres y mujeres. En cuanto a lo de las bebidas, es cierto. Si un cliente está a la espera de una mesa y pasó al bar, puede llevarse su bebida a la mesa si no se la ha terminado cuando se le haya asignado una mesa. A propósito, siempre nos reservamos el derecho de negar el servicio a quien ha consumido demasiado alcohol."

El entrevistador no es sólo un oyente pasivo luego de hacer una pregunta. En este caso, el analista ha conjugado un tema en común a partir de algunas respuestas previas, y ha hecho una pregunta basada en algo que ha salido a colación en algunas ocasiones (la oportunidad de pedir algo de tomar). La respuesta contiene un extracto de la lógica de negocios, una regla que sigue el negocio en una situación en particular. En este caso, la lógica de negocios se aplica a negar el servicio a un cliente alcoholizado.

"Es bueno saberlo. Regresemos a la mesa donde los comensales deciden qué van a consumir."

"Sí. Siempre tenemos algunas sugerencias del día que no están en el menú, y el mesero se las propone a los clientes."

"¿Sabe qué es lo que he visto que pasa? La gente le pide al mesero las recomendaciones, y los meseros por lo general parecen ser sinceros: le dicen si un platillo es mejor que otro. ¿Es algo que usted alienta a hacer?"

"Así es. Ciertamente nuestros meseros comen en el restaurante y tienen sus propias opiniones de lo que les gusta y lo que no. Si a ellos realmente no les gusta un platillo en particular, queremos que lo digan al chef antes que a los clientes, aunque no tengo inconveniente que expresen una preferencia. Claro que no queremos que los meseros le digan a los comensales que la comida es terrible, pero el expresar una preferencia por un platillo no está mal."

"Bien. Vamos a resumir. El cliente o comensal deja sus abrigos, entra al bar, aguarda una mesa, se sienta ante ella, posiblemente ordena una bebida, se le sirve pan y agua y mira el menú."

Se sugiere detenerse y resumir de vez en cuando. Le ayudará a verificar qué tanto ha asimilado y le da la oportunidad de utilizar la terminología del dominio, además de que reconforta al entrevistado saber que usted le ha puesto atención.

www.elsolucionario.net

www.elsolucionario.net

"Así es. El mesero regresa con una bebida y los clientes la beben mientras ven el menú. El mesero les da de cinco a diez minutos para hacer su elección y, luego, regresa. El mesero regresa antes si, claro, los comensales hacen su elección antes."

"¿Cómo saben que deben regresar antes?"

"Bien, los clientes llaman la atención del mesero. Por lo general está cerca del área de la mesa, a menos que haya tenido que ir a la cocina a traer un pedido o necesitara hablar con los chefs por alguna razón."

"¿Área?"

"Sí, a cada mesero se le asigna un área que consta de varias mesas. Hay una sección asignada como área de fumadores, y el resto es para quienes no fuman."

"¿Cómo determina quién atiende un área?"

"Alternamos a los meseros en las diferentes áreas."

"Bueno, volvamos al proceso de servir. Los comensales hacen su elección, el mesero toma la orden en su comanda y luego..."

"Y luego notifica al chef. Esto lo hace mediante la trascripción de la elección en un formulario, llamado comanda, que le da al chef."

"¿Qué hay en la comanda?"

"La mesa, la elección y, muy importante, la hora."

"¿Por qué es tan importante?"

"Debido a que generalmente la cocina está (espero) muy ocupada, y el chef tiene que dar prioridad a las comandas en el orden en que hayan llegado."

"¿Eso es complicado?"

"En realidad, se complica más de acuerdo con su naturaleza."

"¿Cómo está eso?"

"La mayoría de las comidas incluyen un entremés antes del plato principal. A la mayoría de la gente le gusta comer el plato fuerte caliente, así que el chef prepara el entremés, muchos de ellos ya están preparados, como algunas ensaladas, y el mesero se los lleva al cliente. El reto está en llevar el plato fuerte, caliente, a todos los comensales en la mesa al mismo tiempo. Digo 'reto' porque las personas de la mesa por lo general se terminan el entremés en diferentes momentos. Todo el asunto debe estar coordinado."

"Hum. Esto suena a proceso separado. Tomaremos esto en una reunión por separado desde el punto de vista del chef."

www.elsolucionario.net

www.elsolucionario.net

El analista ha tomado una importante decisión: sacar del contexto una secuencia que posiblemente sea parte de un proceso separado. El reconocer en qué momento hacerlo se obtiene mediante la experiencia.

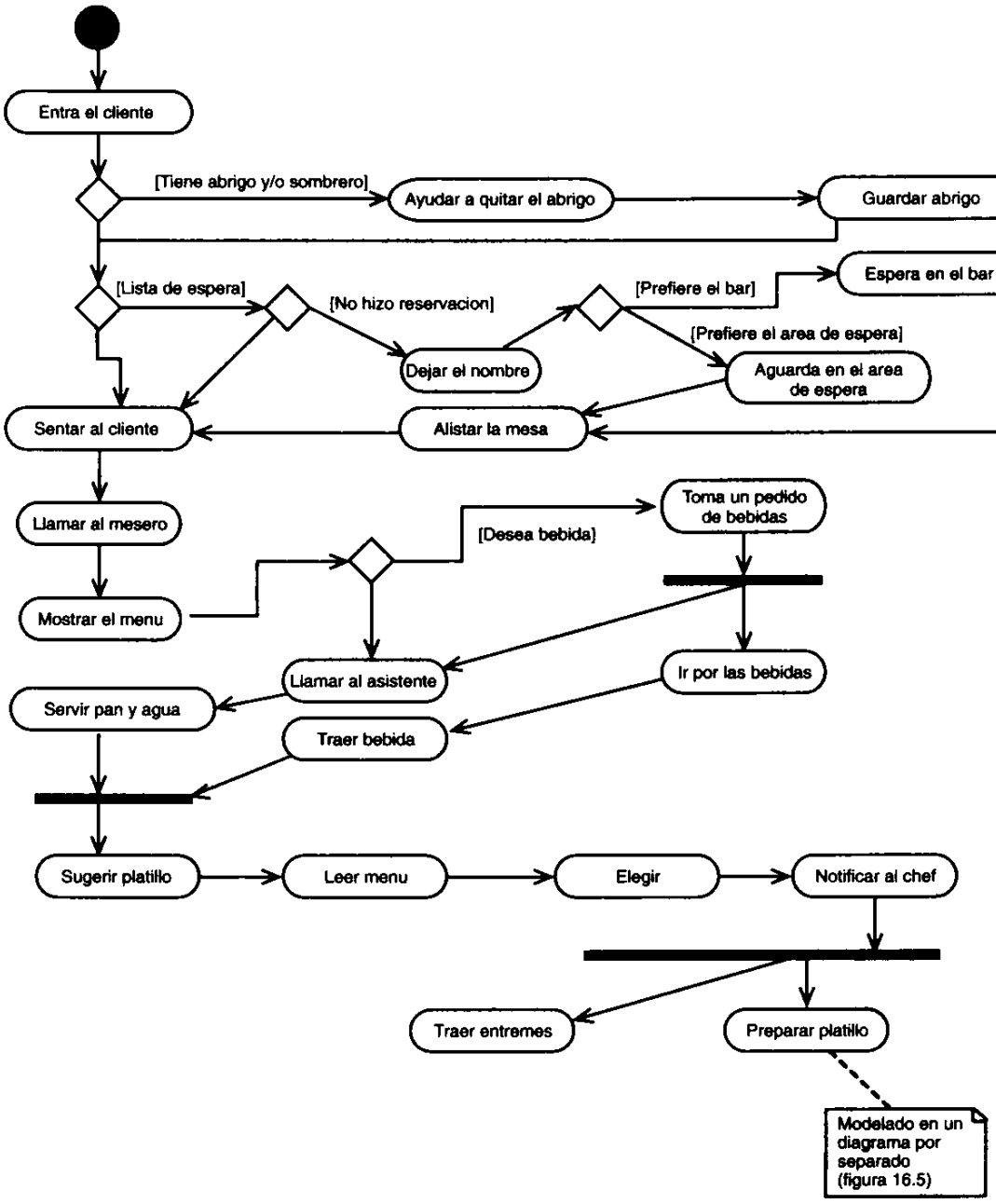
Un buen principio es que, si el entrevistado utiliza palabras como "complejo" o "complicado", o responde que "sí" cuando le pregunta si algo es complicado, tal vez estará ante un conjunto de pasos que necesitarán su propio modelo. Deje que el entrevistado hable un poco antes de tomar una decisión como ésta.

"De acuerdo, es una buena idea."

"Estamos en el punto donde el chef preparará el plato fuerte. A propósito, ¿qué le parece este diagrama?" (Vea la figura 16.2.)

**FIGURA 16.2**

*Las fases intermedias del diagrama de actividades para el proceso del negocio "Servir a un cliente".*




www.elsolucionario.net

www.elsolucionario.net

"Creo que ha comprendido. De cualquier forma, el chef preparará el plato fuerte y el mesero lo recogerá cuando se de cuenta de que los comensales han terminado con el entremés; posteriormente, el mesero llevará el plato fuerte a la mesa. Los comensales ingerirán sus alimentos, y el mesero regresará al menos una vez para verificar si todo está bien."

"Suponga que a un cliente no le ha satisfecho algo de la comida. ¿Qué ocurre?"

"Bueno, hacemos nuestro mejor esfuerzo para que le satisfaga, aun cuando nos cueste algo de dinero. Es mejor perder algo de dinero que a un cliente."

"Buen concepto."

"Gracias. Cuando los comensales terminan con sus alimentos, el mesero regresa y pregunta si desean un postre. En tal caso, el mesero dará el menú de postres y tomará las órdenes. En caso de que no deseen postre, preguntará si desean un café. Si es el caso, traerá café y tazas y les servirá. Si no desean nada, traerá la cuenta. Luego de unos instantes, regresará y obtendrá el pago en efectivo o en tarjeta de crédito. Luego traerá el cambio o los vouchers, los clientes dejarán una propina, recogerán sus abrigos y se irán."

"¿Es todo?"

"No necesariamente. El mesero llamará al mozo de piso para limpiar la mesa, la preparará y estará lista para los siguientes comensales."

"Dado que ello no tiene que ver con el cliente, voy a considerarlo dentro de un proceso por separado, aunque sea breve. Quiero hacerle un par de preguntas. La primera ¿Cómo sabe el mesero que la gente ha terminado?"

"Él permanece en su área, y echa una mirada a cada mesa. La experiencia le dicta cuánto le toma a los comensales ingerir sus alimentos, de forma que se puede anticipar a ello cuando esté cerca de una mesa. ¿Otra pregunta?"

"Sí. Dijo que el mesero podría estar en la cocina para hablar con el chef por alguna razón. ¿Cuál sería tal razón?"

"En ocasiones un cliente necesita saber cuánto tardará la preparación de un alimento. En tales casos, el cliente llama al mesero, quien le pregunta al chef. Una vez que se informa, regresa y le responde al cliente."

Los entrevistadores se aseguran de agregar cualquier pregunta restante al final.

"¿Sabe? Nunca me había dado cuenta de todo lo que ocurre para servir a un cliente en un restaurante."

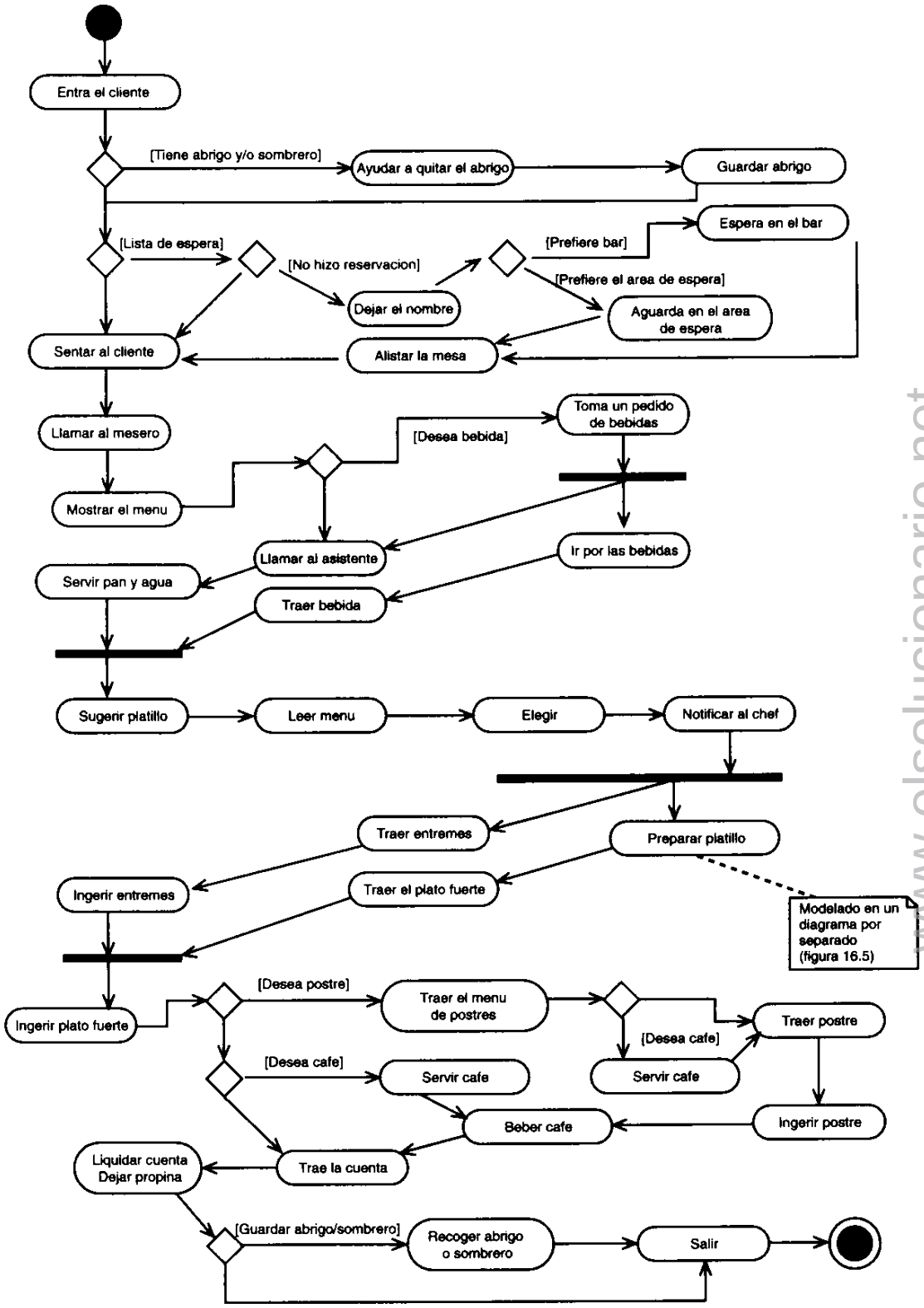
www.elsolucionario.net

www.elsolucionario.net

"Es curioso que lo diga. Hasta que me pidió que le describiera los pasos, yo tampoco lo había analizado mucho. Pienso que su diagrama esquematiza todo lo que he dicho, y es una imagen que aclara mis propias ideas." (Vea la figura 16.3.)

**FIGURA 16.3**

*El diagrama de actividades completo para el proceso del negocio del restaurante, "Servir a un cliente".*

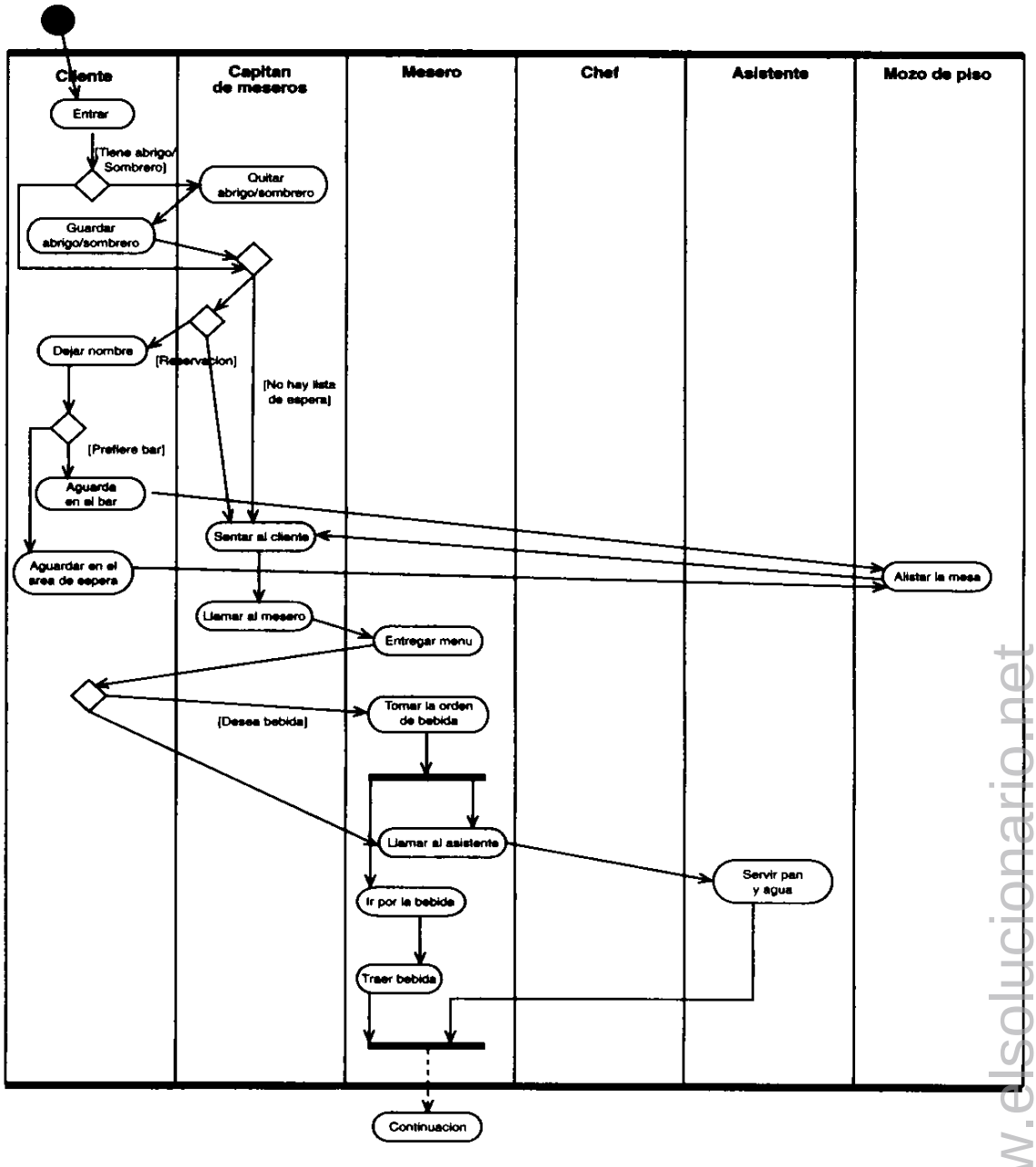



Como lo vio en la hora 11, "Diagramas de actividades", puede convertir un diagrama de éstos en uno con marcos de responsabilidad. Cuando modele un proceso del negocio, será algo muy útil dado que el diagrama con marcos de responsabilidad le muestra la forma en que cada uno de los responsables figuran en el proceso. La figura 16.4 es un diagrama con marcos de responsabilidad para el proceso del negocio "Servir a un cliente".

www.elsolucionario.net

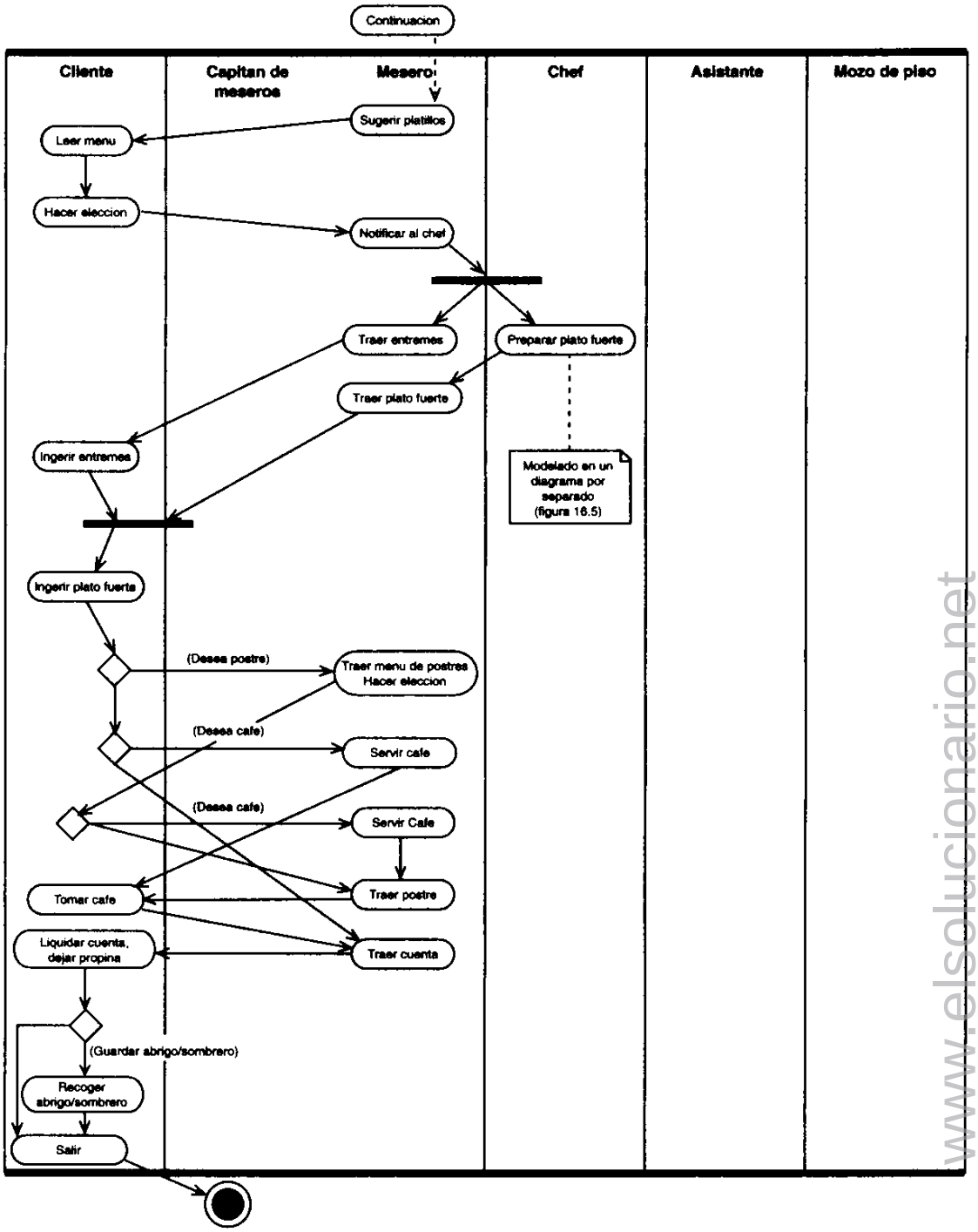
**FIGURA 16.4**

*Un diagrama con marcos de responsabilidad para "Servir a un cliente".*




www.elsolucionario.net

www.elsolucionario.net




www.elsolucionario.net

## Preparación de platillos

¿Recuerda que separó los procesos del negocio que aparecieron en la entrevista? Volvamos a reunir al analista con el restaurantero y exploremos el proceso de la "Preparación de platillos".

"En la plática anterior", decía el analista, "mencionó que muchos platillos incluyen un entremés antes del plato fuerte, y que la mayoría de las personas preferían el plato fuerte caliente. Indicó el reto de traer el plato fuerte a todos al mismo tiempo y tenerlo caliente; y mencionó la importancia de la coordinación. ¿Podría ahondar?"

www.elsolucionario.net

"Claro." dice el restaurantero. "La gente en una mesa casi siempre finaliza sus entremeses, ensaladas o sopas en momentos distintos. Tenemos que coordinarnos para traerles a todos los platos fuertes. La coordinación se realiza entre el mesero y el chef. El chef recibe la comanda del mesero y empieza a preparar los entremeses y el plato fuerte. Cuando se han preparado los entremeses, el mesero va a la cocina, los toma y los lleva a la mesa."

"¿Y el mesero sabe que ya se han preparado los entremeses porque...?"

"Porque va a la cocina de vez en cuando. Es aquí donde se inicia la coordinación: el chef, luego de dar el entremés al mesero, espera a que éste le avise cuando la mayoría de los comensales ya casi ha terminado con sus entremeses para poderle dar el toque final a cada plato fuerte. El mesero permanece en su área designada, y no pierde de vista a la mesa. En el momento adecuado, el mesero va a la cocina, le indica al chef que los comensales casi están listos para el plato fuerte, y el chef termina su preparación. Un chef experto, que cuente con un grupo de asistentes, equilibrará la preparación de platillos para varios comensales a la vez. La meta es tener listo el platillo principal tan pronto como todos los comensales estén listos para él."

"¿Esto siempre ocurre a tiempo?"

"No, no siempre. Pero con un poco de experiencia y sentido común esto podría ser frecuente. Lo que a veces ocurre es que un comensal lento en un grupo no está del todo listo cuando llevamos el platillo fuerte, pero es un problema menor."

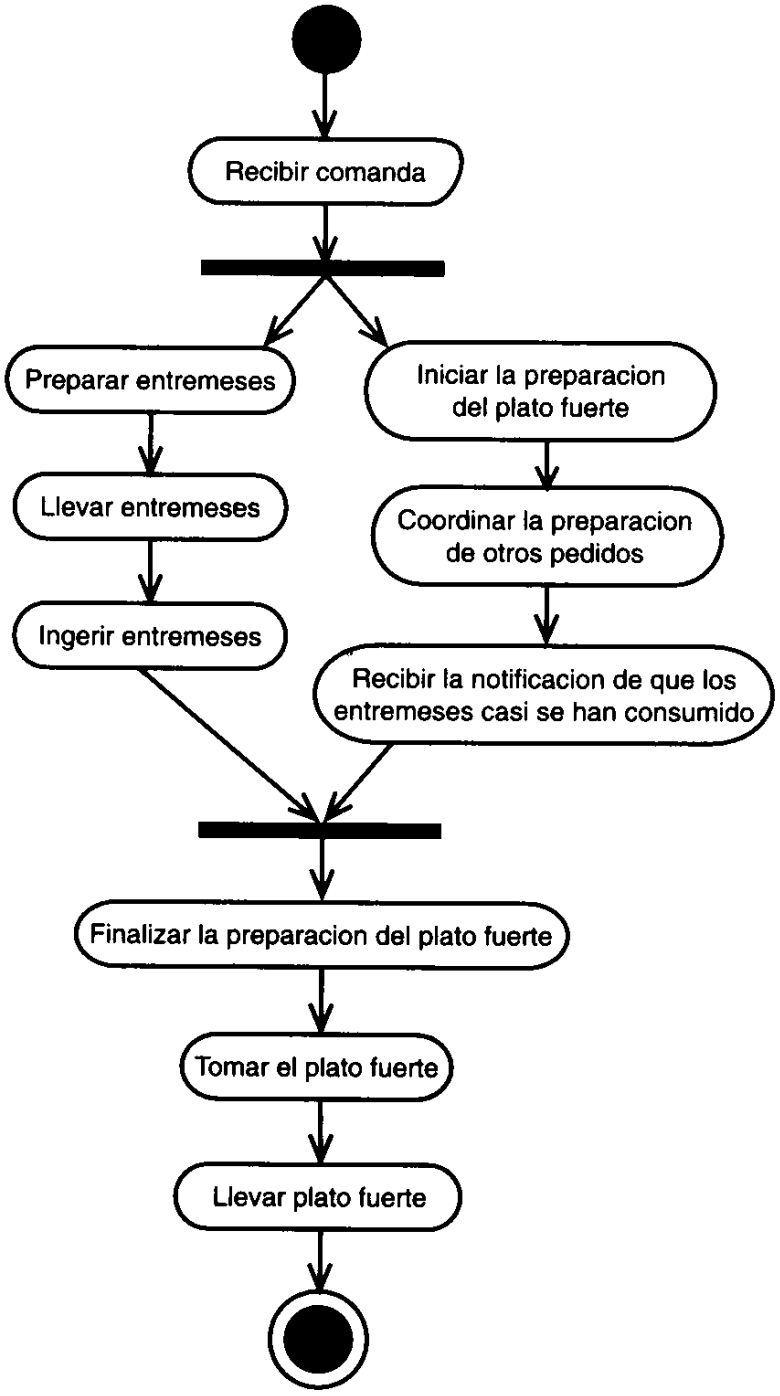
"Ya entiendo. ¿Qué opina de nuestro diagrama para este proceso?" (Vea la figura 16.5.)

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 16.5**

*Un diagrama de actividades para "Preparación de platillos".*



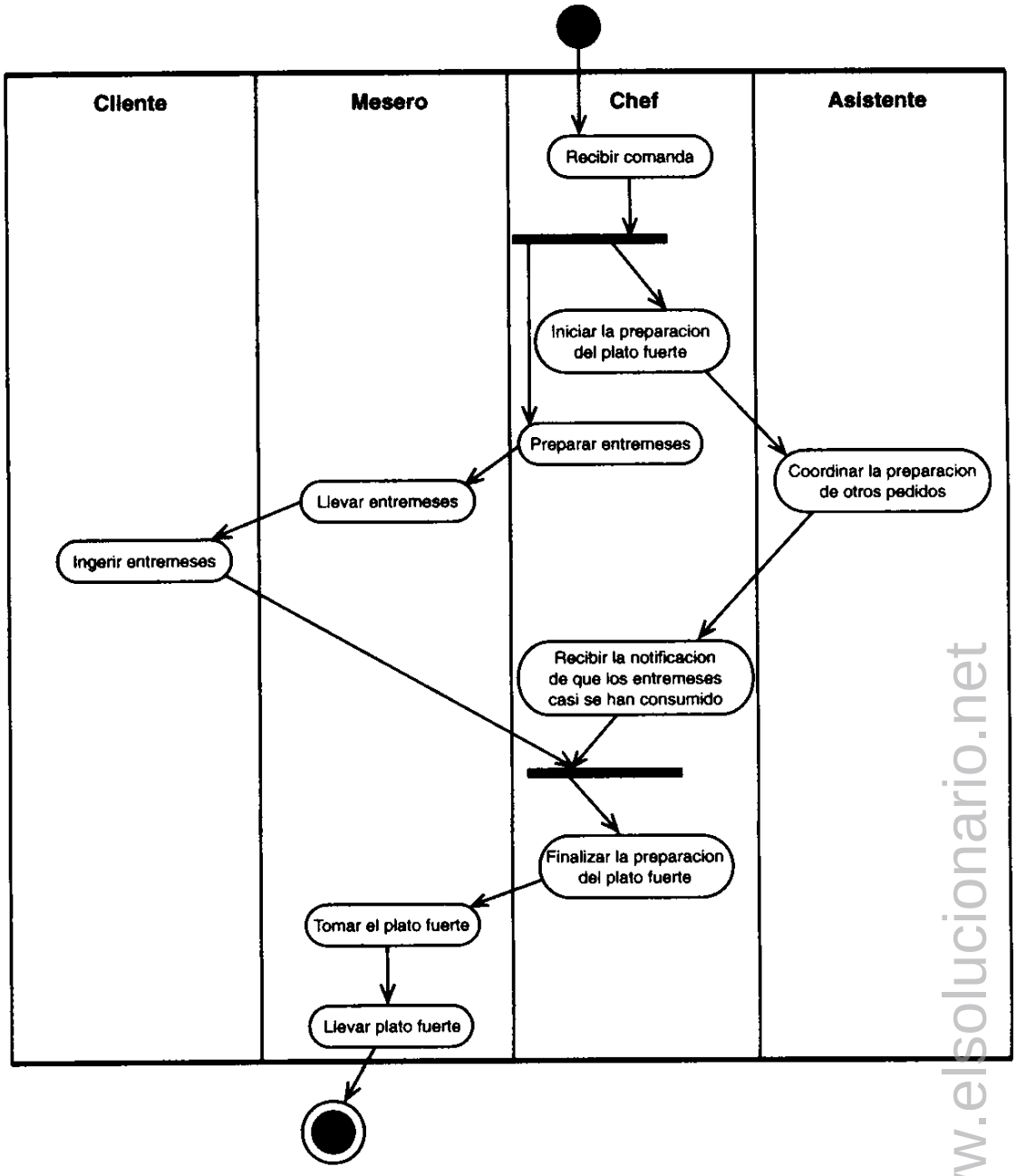

Como en el caso del proceso del negocio anterior, es adecuado generar un marco de responsabilidades, como el de la figura 16.6.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 16.6**

*Un diagrama con marcos de responsabilidades para "Preparación de alimentos".*




## Limpieza de la mesa

"Ahora vayamos a otro proceso por separado, aquél en el que el mozo de piso limpia la mesa", dice el analista.

"Éste también requiere cierta coordinación. El mesero se asegura de que todos se hayan ido y, luego, llamará al mozo de piso para que se encargue de la mesa. En una noche ajetreada, esto tendrá que hacerse con rapidez. No tenemos tantos mozos de piso como meseros, dado que esto es un proceso casual. Los mozos de piso no siempre están cerca, por lo que tal vez el mesero se vería en la necesidad de buscar uno."

"Creo comprenderlo con 'encargarse de la mesa', pero ¿podría ser un poco más específico?"

www.elsolucionario.net

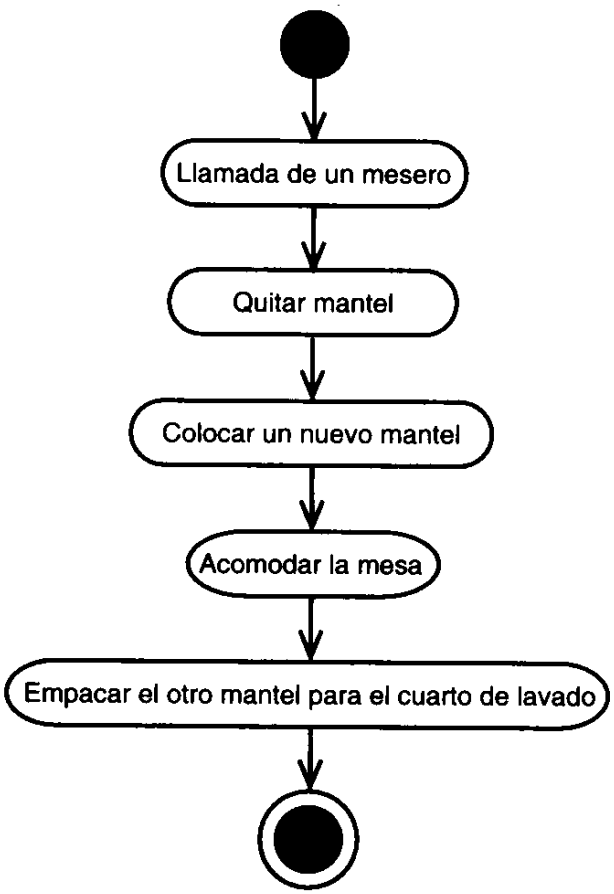
www.elsolucionario.net

"Claro. En los restaurantes que yo manejo, tenemos un mantel para cada servicio. Así que el mozo de piso tiene que quitar el mantel usado, amarrarlo, colocar uno nuevo y un nuevo juego de cubiertos en la mesa. Dobla las servilletas y acomoda los cubiertos y un plato para cada asiento de la mesa. Luego se lleva el mantel doblado a una habitación detrás de la cocina. Lo empacamos y lo enviamos al cuarto de lavado al día siguiente."

La figura 16.7 muestra el diagrama de actividades para este proceso.

**FIGURA 16.7**

*Un diagrama de actividades para "Limpieza de la mesa".*




# Lecciones aprendidas

Si usted es aspirante a analista, recuerde las lecciones aprendidas en esta entrevista:

- Se recomienda detenerse y resumir de vez en cuando para verificar su asimilación, practicar con la terminología y hacer que el entrevistado se sienta cómodo.
- Siempre pida que el entrevistado le explique cualquier terminología que no le sea familiar. No se preocupe si ello aparenta falta de conocimiento. La razón por la que está aquí es para adquirirlo y aprender la terminología. Después de todo, va a tener que utilizar el nuevo vocabulario cuando penetre en el análisis del dominio.
- A veces, podrá hacer una pregunta basada en un tema que perciba como respuesta a preguntas anteriores. Mantenga su mente y oídos abiertos para hacer preguntas como éstas. La lógica de negocios con frecuencia emerge de las respuestas.

www.elsolucionario.net

www.elsolucionario.net

- Tome nota cuando aparezcan las reglas de la lógica de negocios. Lleve un registro de estas reglas. Podrían ser útiles posteriormente (usted nunca sabe: algún día podría generar una herramienta para la automatización de decisiones que dependa de tales reglas). Claro que debe generarse una registro de esto en la minuta.
- Si siente que cierta parte del proceso se torna complejo, puede optar por sacar tal complicación del contexto hacia un proceso del negocio por separado. Podría facilitar el modelado y el modelo resultante será más claro que si intenta aglomerarlo todo en un solo proceso.
- Obtenga el punto de vista del entrevistado respecto al diagrama de actividades. Haga cualquier modificación que sugiera.

Ha aprendido mucho esta hora y ahora ya cuenta con algunas valiosas técnicas. Conforme obtenga experiencia, generará sus propias técnicas.

En la siguiente hora, comprenderá el análisis de dominios.

# Resumen

Esta hora le presentó el escenario para el estudio de un caso que aplicará al UML en un proceso de desarrollo. En el escenario, el consorcio ficticio LaHudra, Nar y Goniff, S. A. decide incorporar la tecnología del cómputo en el restaurante del futuro. Como analista, su trabajo es comprender el proceso del negocio involucrado, comprender el dominio y recopilar las necesidades (acciones presentes en el primer segmento de GRAPPLE).

La recién creada División de restaurantes LNG le da acceso a los expertos del dominio que necesita para comprender los procesos del negocio.

El contenido de esta hora se ha orientado al diálogo en una entrevista y la forma en que podría fluir. Se han intercalado notas que le servirán de guía para saber cómo realizar la entrevista. El objetivo fue el de mostrarle cómo convertir los resultados de la entrevista en un modelo UML.

En la siguiente hora, comprenderá lo relacionado al análisis de un dominio.

www.elsolucionario.net

www.elsolucionario.net

# Preguntas y respuestas

**P ¿Siempre se da el caso en que las acciones de un segmento se realicen en el orden en que las listó?**

**R** No. En ocasiones podría ser sensato realizarlas en otro orden. Por ejemplo: tal vez quiera descubrir los requerimientos de un sistema antes de identificar a los elementos cooperativos. A su vez, tenga en cuenta que algunas de las acciones no siempre serán necesarias para ciertos proyectos y que algunas de ellas pueden realizarse al mismo tiempo que otras. La "G" en GRAPPLE significa "Guidelines" o "Lineamientos". Ello no se puede interpretar como "¡Gulp! Es una obligación seguirlo así" o "Debo hacerlo exactamente así".

**P ¿Es necesario contar con un solo entrevistador para enterarse de los procesos de negocios de un cliente o un experto? ¿Dos funcionarían mejor que uno?**

**R** Por lo general se recomienda que sea una sola persona la que hable con el experto, para que el entrevistado no se sienta como que encara a la inquisición. Podría optar por rotar a los entrevistadores en algún momento de una sesión. El segundo entrevistador podría haber sido el que tomaba las notas, mismo que alternaría su papel con el primer entrevistador.

**P ¿Hay alguna consideración especial para las notas de la entrevista?**

**R** Asegúrese que haya indicado la fecha, hora, lugar y participantes al principio. Nunca sabrá cuándo necesitará tal información y no necesitará tenerla de memoria. A su vez, dentro de las notas intente capturar tanto como le sea posible. Es como un estenógrafo en una corte. Si intenta hacer un boceto conforme avanza, podría estar pasando por alto algo.

**P ¿No podría omitir algo al intentar tomar nota de todo?**

**R** ¡Por supuesto —razón por la cual es mejor contar con dos personas que tomen notas—! Seguramente uno de ellos podría haber anotado algo que el otro no. Recuerde que las notas que tome serán parte del documento que le dará al cliente. Entre más completas estén las notas, será más fácil rastrear la evolución de una idea.

# Taller

Para realmente captar todo esto, responda al cuestionario y haga los ejercicios. Las respuestas están en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿Cuál diagrama del UML es adecuado para modelar un proceso del negocio?
2. ¿Cómo podría modificar a este diagrama para mostrar qué hace cada quién?
3. ¿Qué debe entenderse por "lógica de negocios"?

www.elsolucionario.net

www.elsolucionario.net

## Ejercicios

1. Intente aplicar los principios de esta hora a otro dominio. Suponga que LaHudra, Nar y Goniff le han contratado para encabezar un equipo de desarrollo que genere un sistema para su biblioteca corporativa. Inicie el segmento Recopilación de necesidades mediante la asimilación y modelado de los procesos del negocio involucrado. Para ello, tendrá que confiar en su propio conocimiento de las bibliotecas. Haga anotaciones de su solución dado que utilizará este ejemplo en los ejercicios de las siguientes horas.
2. Revise las entrevistas de esta hora. ¿Qué partes de la lógica de negocios se hicieron evidentes?

www.elsolucionario.net

www.elsolucionario.net

# HORA 17

# Elaboración de un análisis de dominio

Ahora continuaremos con el análisis conceptual en el segmento de recopilación de necesidades de GRAPPLE.

En esta hora se tratarán los siguientes temas:

- Análisis de la entrevista del proceso del negocio
- Desarrollo del diagrama de clases inicial
- Agrupación de las clases
- Conformación de asociaciones
- Agregados y objetos compuestos
- Llenado de clases las clases

El primer par de acciones en GRAPPLE se orientan al dominio y no al sistema. Ni la hora anterior ni ésta se orientarán al sistema propuesto. Ciertamente, en lo que hemos visto hasta ahora no se ha propuesto sistema alguno. Sólo tenemos una asignación vaga de LaHudra, Nar y Goniff para valernos de la tecnología y mejorar el acto de comer en restaurantes.

www.elsolucionario.net

www.elsolucionario.net

El objetivo de las horas 16 y 17 es comprender el dominio. Lo que significa que tenemos que conocer los procesos específicos que intentamos mejorar y el modo en el que operan tales procesos. Al intentar descubrir los procesos del negocio, hemos empezado a alimentar el conocimiento del equipo de desarrollo en nuestro escenario. Como resultado, los miembros del equipo cuentan con un vocabulario que pueden utilizar para comunicarse posteriormente con la División de Restaurantes de LNG. Esto es de gran importancia dado que el equipo cuenta con un fundamento para aumentar y cultivar su conocimiento en el transcurso del proyecto.

# Análisis de la entrevista del proceso del negocio

El equipo de desarrollo tendrá otras entrevistas con los expertos restauranteros, pero antes de ello trabajarán con el contexto de la entrevista del proceso del negocio. El objetivo es producir un diagrama de clases inicial. Un modelador de objetos hará este trabajo con el equipo durante la entrevista o mediante los resultados de ella. En este punto, el modelador buscará sustantivos, verbos y construcciones verbales. Algunos de los sustantivos se convertirán en clases dentro del modelo, y algunos otros en atributos. Los verbos y construcciones verbales podrán convertirse en operaciones o las etiquetas de asociaciones.

Examinemos los resultados de la entrevista de la hora anterior. ¿Qué sustantivos y verbos utilizó el restaurantero?

He aquí los sustantivos:

cliente, abrigo, guardarropa, vale de guardarropa, sombrero, línea de espera, lista de espera, reservación, nombre, bar, bebida, comida, área de espera, mesa, mozo de piso, mantel, capitán de meseros, mesero, servidor, área de servicio, comensal, menú, asistente, charola, pan, mantequilla, vaso, agua, persona, elección de menú, sugerencia del día, restaurante, chef, platillo, cocina, comanda, área de fumadores, formulario, hora, entremés, plato fuerte, postre, menú de postres, café, taza, cuenta, efectivo, tarjeta de crédito, cambio, voucher, propina, cubierto, servilleta, habitación, cuarto de lavado.

Observe que hemos utilizado cada sustantivo en su forma singular.

Los verbos y construcciones verbales son:

tener, ayudar, almacenar, dar, formarse, honrar, sentar, salir, aguardar, surgir, deshacerse, establecer, caminar, llamar, rondar, ver, gesticular, mostrar, preguntar, ordenar, decidir, traer, ir, obtener, finalizar, reservar, rehusar, relatar, pedir, recomendar, alentar, gustar, decir, expresar, mirar, regresar, beber, leer, permitir, seleccionar, atender, obtener un

www.elsolucionario.net

www.elsolucionario.net

pedido, servir, recolectar, dejar, limpiar, alistar, anticipar, hablar, venir, convocar, localizar, proveer, preferir, coordinar, recibir, verificar, depender, cuidar, limpiar, asegurarse, encargarse, buscar, quitar, atar, doblar, arreglar, empacar, enviar.

Cuando busquemos los sustantivos y verbos, deberemos procurar incluirlos todos. ¿El modelador los incluirá a todos en el modelo? No. El sentido común indicará a cuáles incluir y a cuáles no. Será de mucha ayuda tener mayor contacto con el restaurantero.

# Desarrollo del diagrama de clases inicial

Ahora pongámonos en los zapatos del modelador y empecemos a desarrollar el diagrama de clases. Es aquí donde el sentido común entra en juego. Primero eliminaremos algunos de los sustantivos.

Recordemos en la entrevista que se intentó cambiar "mesero" por "servidor", pero que no dio resultado. Así que podríamos eliminar uno de estos términos. Tanto el entrevistado como el entrevistador decidieron usar "mesero", así que eliminaremos a "servidor". "Cliente" y "comensal" son sinónimos, por lo que podremos eliminar otro sustantivo: nos quedaremos con "cliente". "Persona" parece ser demasiado genérico, así que también lo eliminaremos.

¿Podríamos eliminar a otros? Algunos sustantivos son más adecuados para usarse como atributos y no como clases. En nuestra lista, "nombre", "hora" y "reservación" caen en tal categoría. Otro de ellos, "cuarto de lavado", no es parte física del restaurante, por lo que podríamos eliminarlo.

He aquí el otro lado de la moneda: también es posible agregar una o dos clases. Si analizamos la entrevista, veremos que el restaurantero hizo referencia a "áreas asignadas" y "rotación de meseros". ¿Quién hace la "asignación" y la "rotación"? Pues un "gerente", que agregaremos a la lista. Tal clase podría no haber surgido de la entrevista original, dado que el analista se enfocó al cliente, el mesero, el chef y el mozo de piso.

La adición de una clase (como lo verá en la adición de clases abstractas) refleja la evolución del entendimiento conforme avanza el proyecto.

www.elsolucionario.net

www.elsolucionario.net

Luego de filtrar los sinónimos y atributos, así como agregar una clase, ahora ésta será nuestra lista de sustantivos que se convertirán en clases:

cliente, abrigo, guardarropa, vale de guardarropa, sombrero, línea de espera, lista de espera, bar, bebida, comida, área de espera, mesa, mozo de piso, mantel, capitán de meseros, área de servicio, menú, asistente, charola, pan, mantequilla, vaso, agua, servicio, mesero, elección, sugerencia del día, restaurante, chef, platillo, cocina, comanda, área de fumadores, formulario, entremés, plato fuerte, postre, menú de postres, café, taza, cuenta, efectivo, tarjeta de crédito, cambio, voucher, propina, cubierto, servilleta, habitación, reservación, gerente.

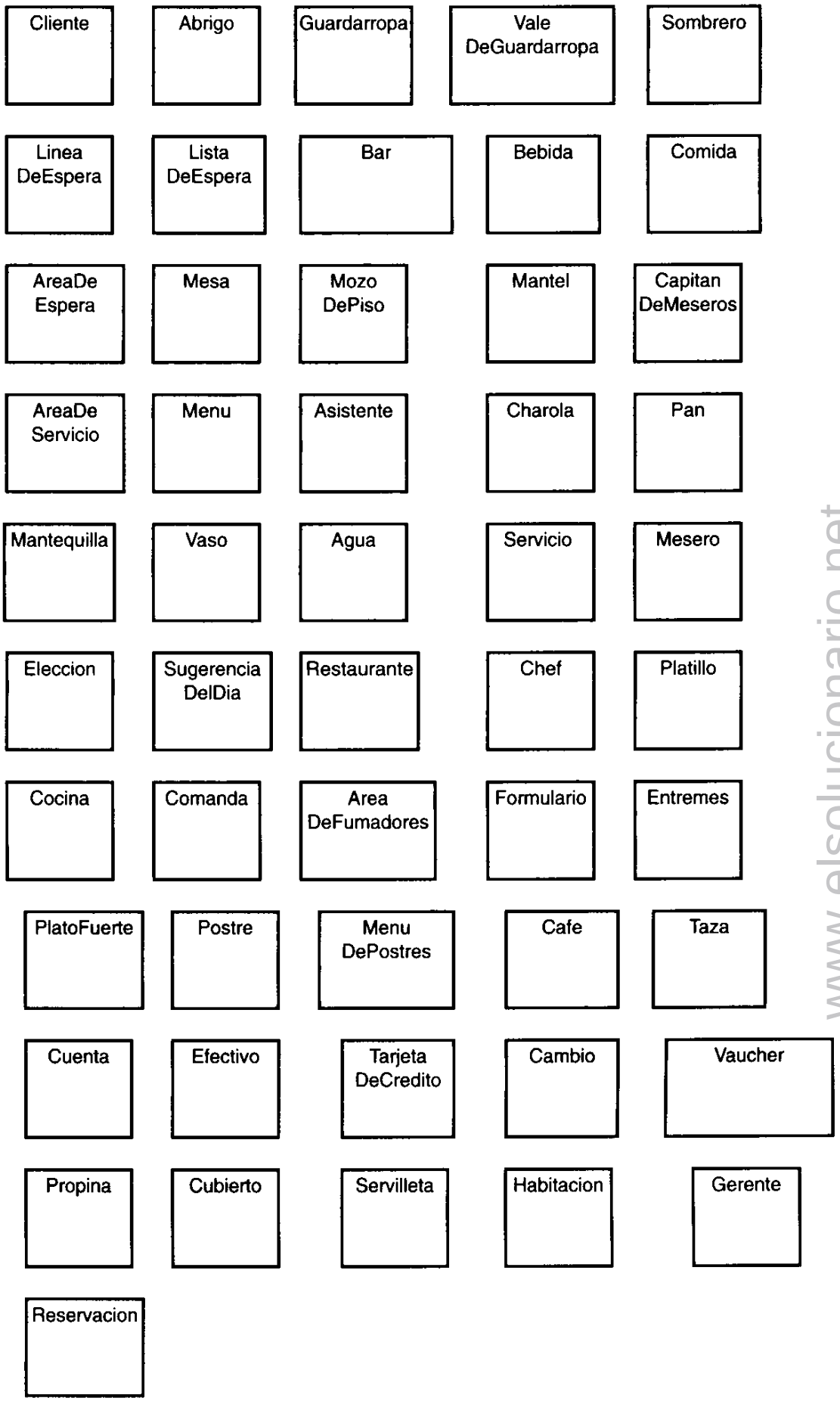
Nos serviremos de estos sustantivos para generar el diagrama de clases de la figura 17.1, donde pondremos en mayúscula la primera letra de cada nombre de clase. Si tal nombre tiene más de una palabra, las uniremos y pondremos con mayúscula la primera letra de cada palabra. También eliminaremos los acentos.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 17.1**

*El diagrama de clases inicial para el dominio del restaurante.*




www.elsolucionario.net

www.elsolucionario.net

# Agrupación de las clases

Ahora intentaremos conformar algunos grupos significativos. Uno de los grupos consta de personas: cliente, servicio, mozo de piso, capitán de meseros, asistente, chef, mesero y gerente. Este grupo podría tener alguna división pues todos, excepto el cliente y el servicio, son empleados. Así que nos quedaremos con los grupos cliente, servicio y empleado.

Otro grupo consta de elementos relacionados con alimentos: bebida, comida, pan, mantequilla, agua, sugerencia del día, platillo, entremés, plato fuerte, postre y café.

Hay un tercer grupo que consta de utensilios: vaso, cubierto, charola, servilleta y mantel.

El cuarto grupo contiene elementos de transacción: vale de guardarropa, cuenta, efectivo, cambio, tarjeta de crédito, pagaré y propina.

Existe otro grupo que consta de áreas del restaurante: área de espera, área de fumadores, bar, guardarropa, cocina, área de servicio, mesa y habitación. "Habitación" se refiere a aquella que almacena los manteles (y asumimos que otros elementos) que el restaurante enviará a la lavandería. Para hacerlo más descriptivo, llamémoslo "cuarto de lavado".

Finalmente, podemos agrupar los formularios del restaurante: menú, menú de postres, vale de guardarropa, cuenta y formulario. El último se refiere al que se entrega al chef con la orden, por lo que lo llamaremos "comanda".

Observe que un par de estos elementos pueden encontrarse en dos grupos (formularios y elementos de transacción). Esto, como veremos, es admisible.

¿Ahora qué haremos con tales grupos? Cada nombre de grupo puede convertirse en una clase abstracta: una que no genera instancias por sí misma, pero que funciona como una clase principal de clases secundarias. Así, la clase abstracta AreaDeRestaurante tiene las siguientes clases secundarias: Bar, AreaDeServicio, Mesa, AreaDeEspera, Guardarropa y Cocina.

Podemos modificar el diagrama de clases de la figura 17.1 y producir el de la figura 17.2.

www.elsolucionario.net

www.elsolucionario.net

# Conformación de asociaciones

Ahora, crearemos y etiquetaremos asociaciones entre algunas de las clases. Los verbos y las construcciones verbales podrán ayudarnos con las etiquetas, y no debemos limitarnos sólo a las de la entrevista. Las etiquetas que sean más descriptivas podrían sobreentenderse.

Una estrategia es la de enfocarnos en algunas de las clases, ver cómo se asocian entre sí, e ir a otro grupo hasta que hayamos desmadejado al conjunto de clases. Posteriormente, generaremos las agregaciones y objetos compuestos. A continuación, incorporaremos los verbos y construcciones verbales como operaciones de las clases.

## Asociaciones con el cliente

Empecemos con la clase Cliente. ¿Cuáles clases se asocian con ella? La Reservacion sería una de ellas, y Mesero otra. Algunas otras serían Menu, Alimento, MenuDePostre, Postre, Orden, Cuenta, Propina, Abrigo y Sombrero. La figura 17.3 muestra las asociaciones.

En este punto podemos tomar algunas decisiones. ¿Es necesario incluir Sombrero y Abrigo? Después de todo, nos enfocamos en servir un alimento. Luego de debatir, probablemente concluiríamos que tales clases podrían quedarse en el modelo, porque nuestro interés se centra en todo el proceso de salir a comer. Esto nos hará generar otra clase, EncargadoDelGuardarropa, dado que alguien deberá guardar el abrigo y el sombrero del cliente.

Vamos a etiquetar las asociaciones con algunas frases que las distingan. He aquí algunas de ellas:

- El Cliente hace una Reservacion
- El Cliente es atendido por un Mesero
- El Cliente ingiere un Alimento
- El Cliente ingiere un Postre
- El Cliente hace una Orden
- El Cliente elige de un Menu
- El Cliente elige de un MenuDePostre
- El Cliente liquida la Cuenta
- El Cliente deja una Propina
- El Cliente da a guardar un Abrigo a un EncargadoDelGuardarropa
- El Cliente da a guardar un Sombrero a un EncargadoDelGuardarropa

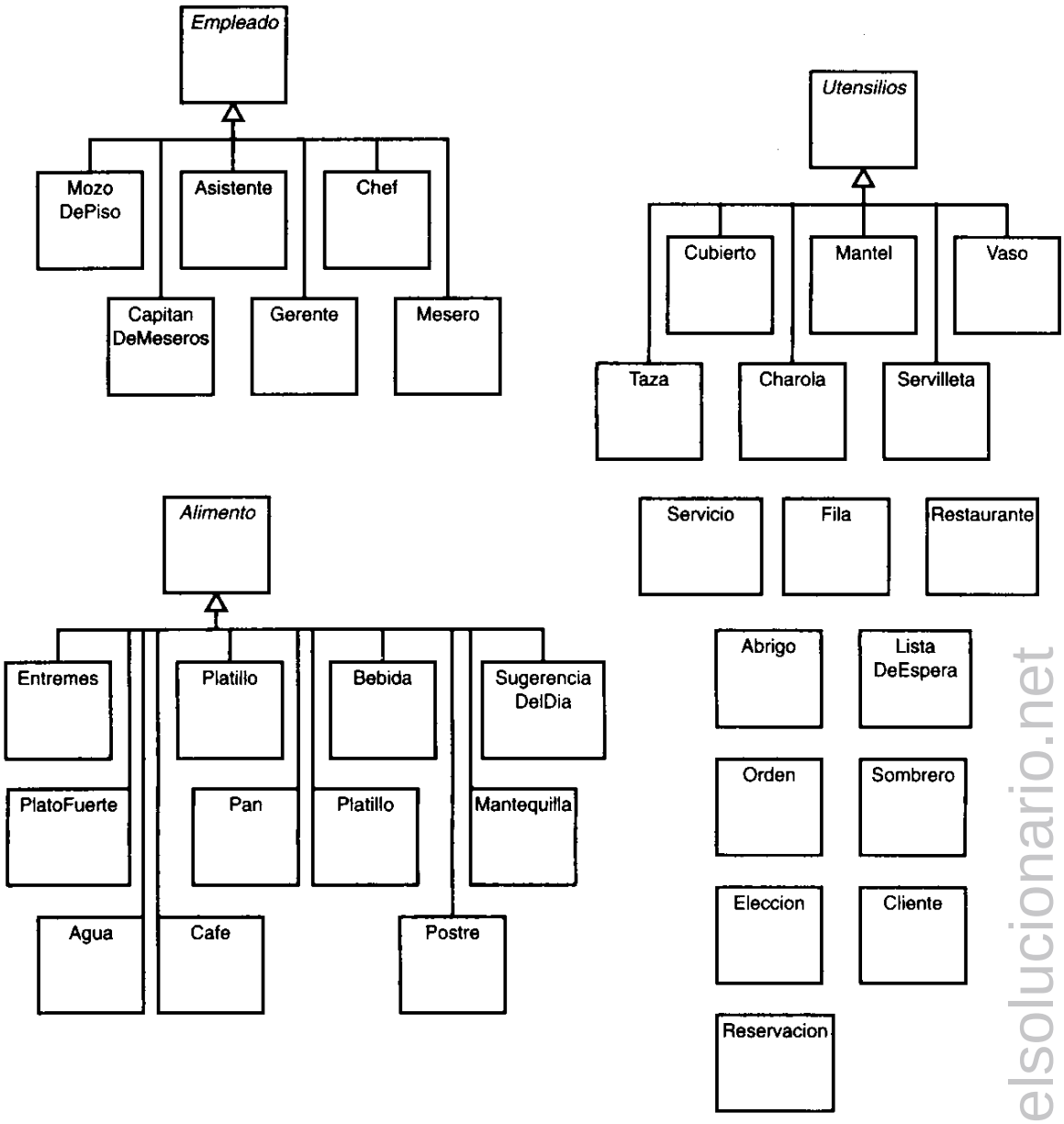
La figura 17.4 le muestra las asociaciones etiquetadas.

www.elsolucionario.net

www.elsolucionario.net

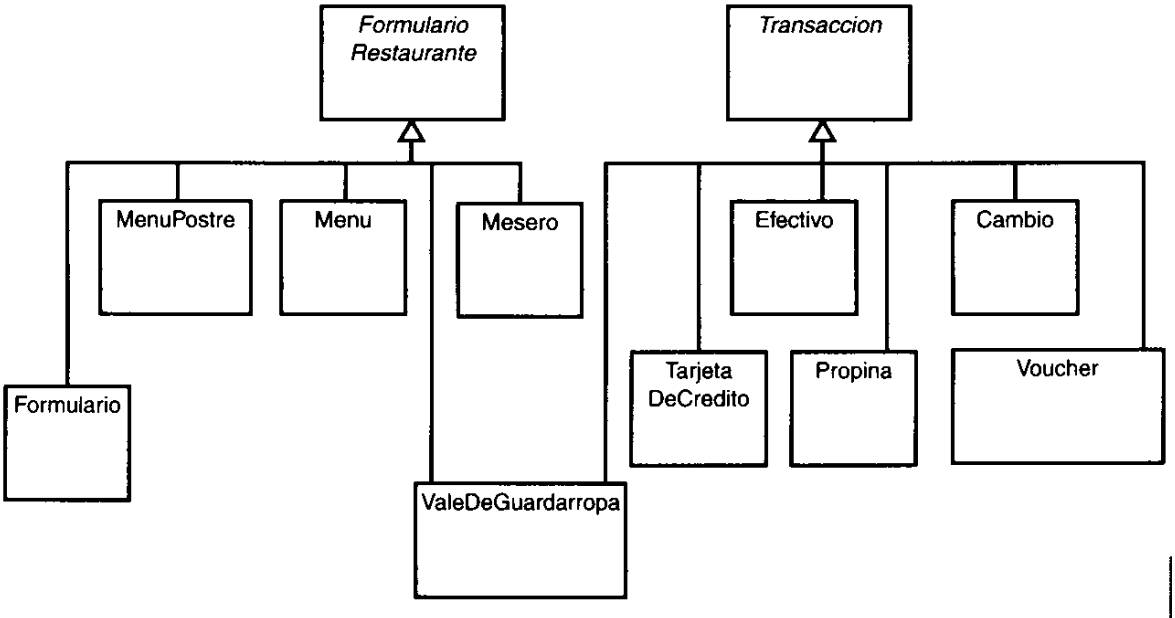
**FIGURA 17.2**

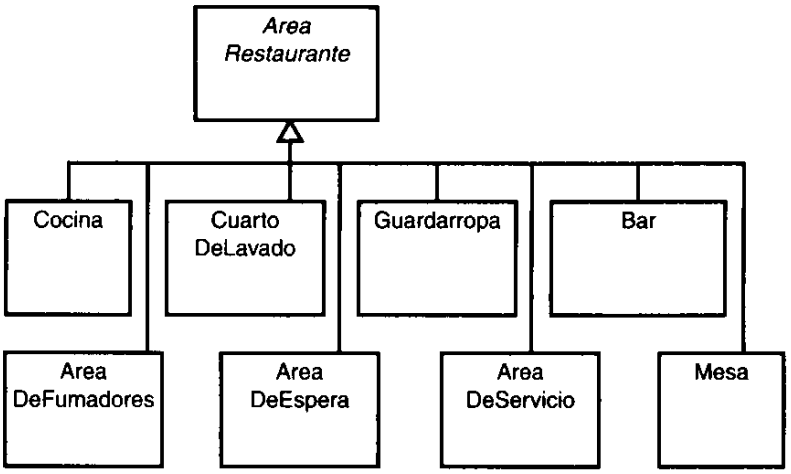
*Las clases abstractas dividen al diagrama de clases en grupos significativos.*




www.elsolucionario.net

www.elsolucionario.net




www.elsolucionario.net

www.elsolucionario.net

**FIGURA 17.3**

*Las asociaciones iniciales a la clase Cliente.*

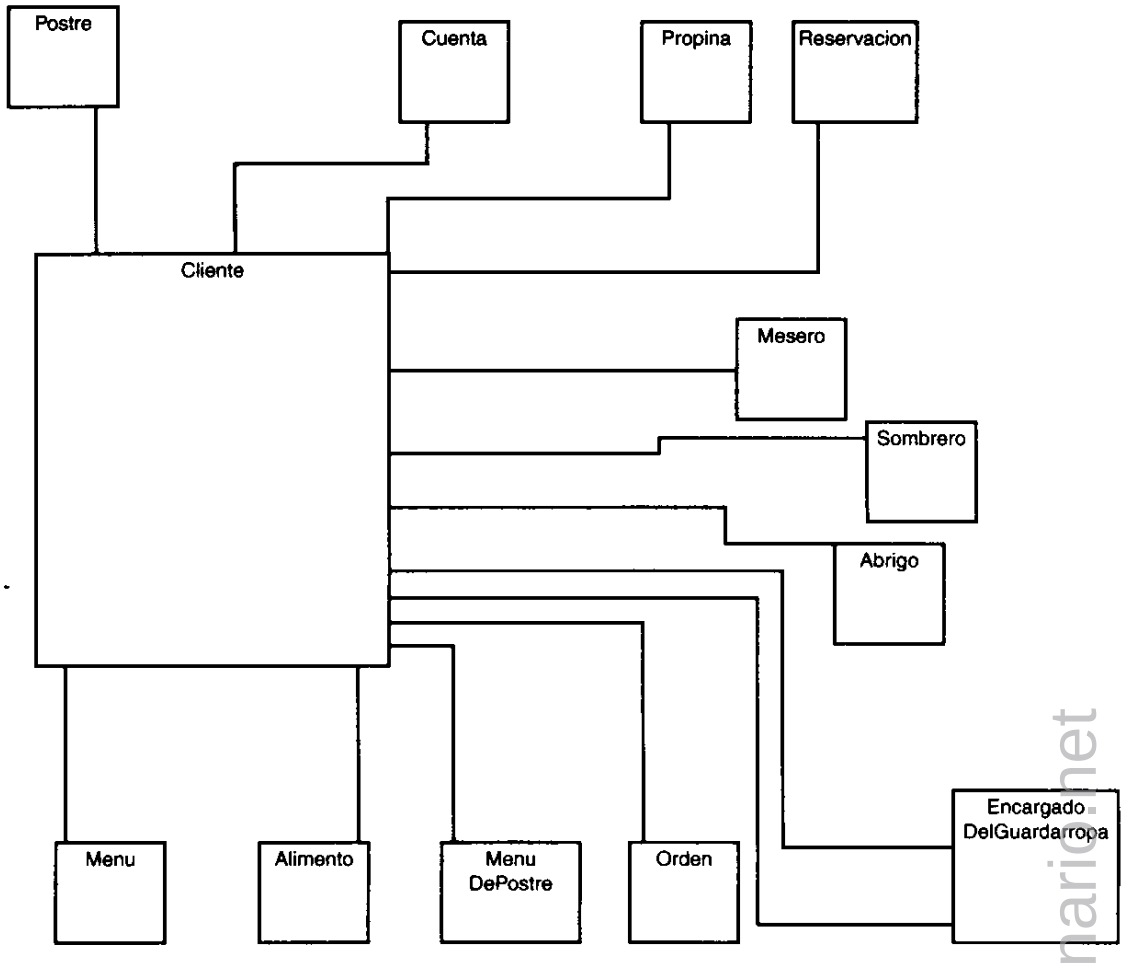
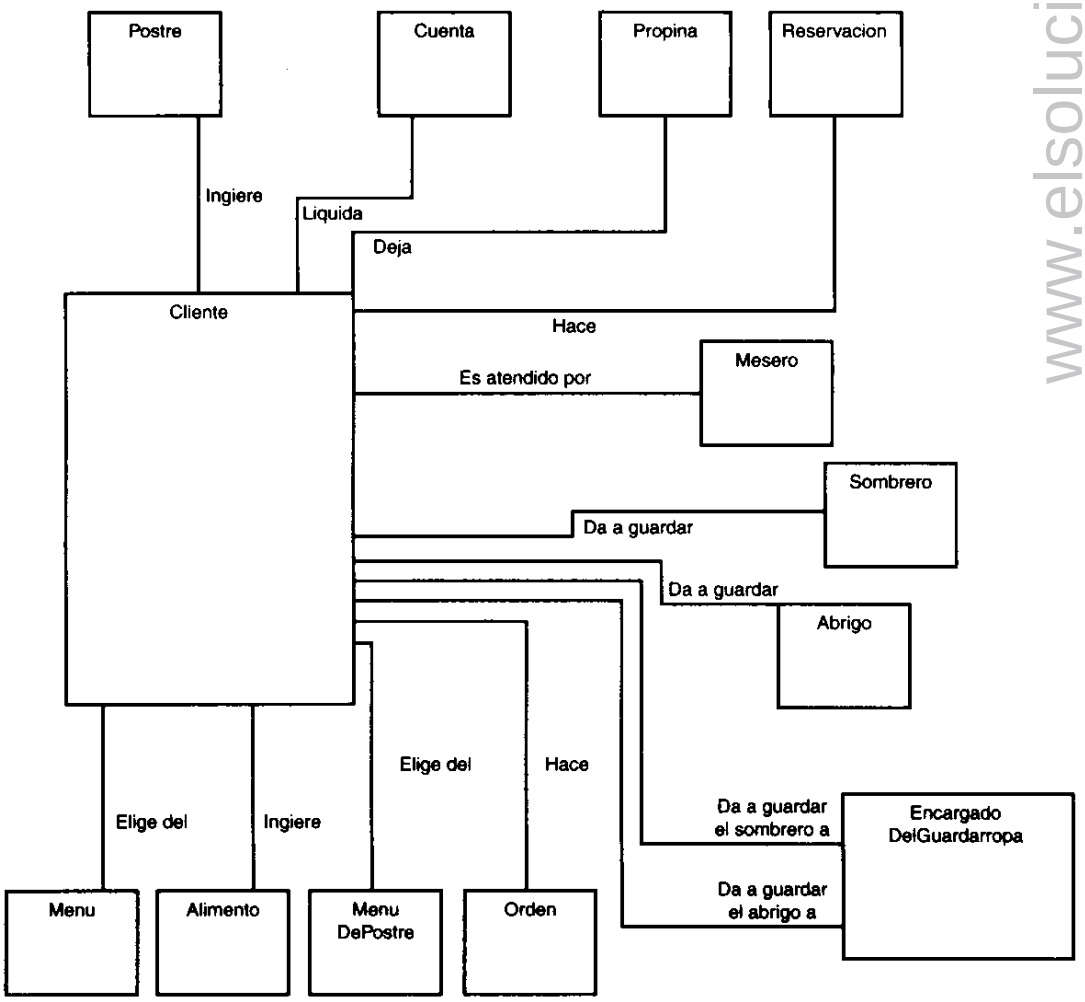



**FIGURA 17.4**

*Las asociaciones a la clase Cliente rotuladas.*




www.elsolucionario.net

www.elsolucionario.net

Ahora vamos a enfocarnos en las multiplicidades. Recuerde que una multiplicidad es parte de una asociación: indica cuántas instancias de la clase B se asocian con una de la clase A.

En la mayoría de las frases enumeradas, el Cliente se relaciona con una instancia de otra clase. La segunda frase es distinta de las otras. Tiene una voz pasiva ("es atendido por") en lugar de una voz activa de otras (como "liquida" o "deja"). Esto sugiere que algo diferente podría ocurrir con tal asociación. Si la invertimos y examinamos la asociación desde el punto de vista del mesero ("El Mesero sirve a un Cliente"), es evidente que un Mesero puede atender a varios clientes.

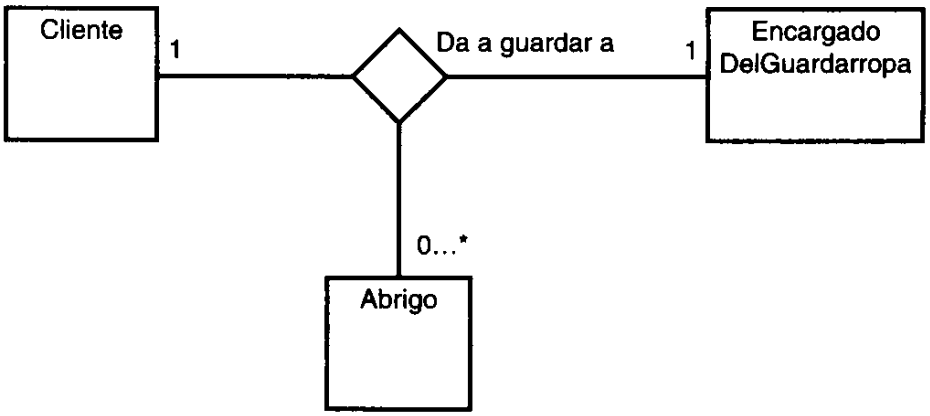
Las dos frases finales apuntan a un tipo de asociación que no habíamos visto antes:

- El Cliente da a guardar un Abrigo a un EncargadoDelGuardarropa
- El Cliente da a guardar un Sombrero a un EncargadoDelGuardarropa

¿Cómo modelaríamos esto?

A este tipo de asociación se le conoce como *tripartita*; esto quiere decir que hay tres clases involucradas. Este tipo de asociación la modelaría mediante la conexión de las clases asociadas con un rombo, y escribiría el nombre de la asociación cerca de él, como en la figura 17.5. En una asociación tripartita, las multiplicidades indican cuántas instancias de dos clases están involucradas cuando la tercera se mantiene constante. En este ejemplo, un Cliente puede dar a guardar más de un Abrigo a un EncargadoDelGuardarropa.

**FIGURA 17.5**
*Una asociación ternaria.*




En la siguiente subsección, verá otra forma de manejar esto.

Es posible contar con más de tres clases en una asociación. En nombre de la generalización, el UML la trata como asociación múltiple.

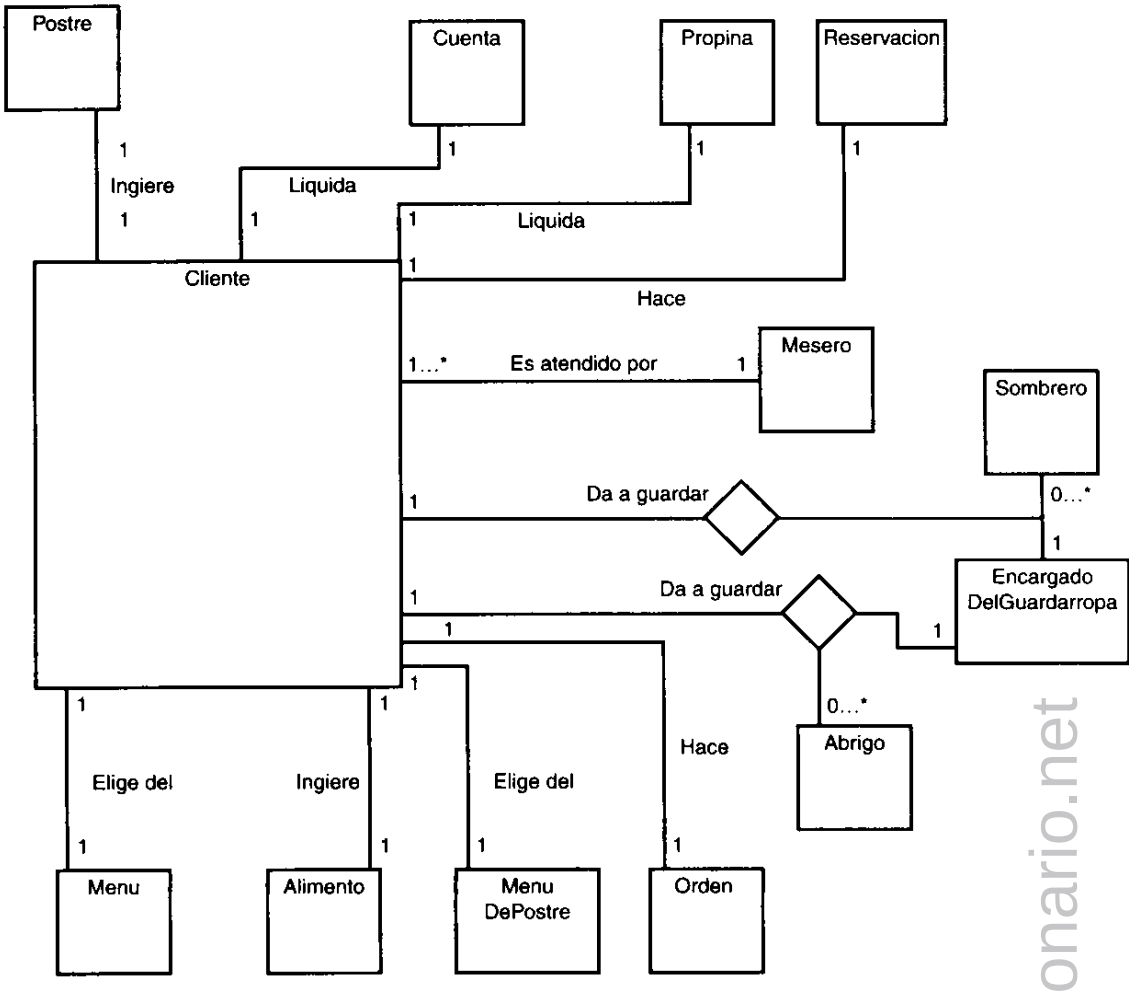
La figura 17.6 le muestra todas las asociaciones del Cliente etiquetadas, donde se incluyen las multiplicidades.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 17.6**

*Inclusión de las multiplicidades en las asociaciones de la clase Cliente.*




## Asociaciones con el Mesero

Ahora utilicemos la asociación Cliente-Mesero para generar las asociaciones con el Mesero. Una forma de modelar cualquiera de las asociaciones del mesero es tratarlas como tripartitas:

- El Mesero toma una Orden de un Cliente
- El Mesero lleva una Orden a un Chef
- El Mesero sirve un Alimento a un Cliente
- El Mesero sirve un Postre a un Cliente
- El Mesero trae un Menu a un Cliente
- El Mesero trae un MenuDePostre a un Cliente
- El Mesero trae la Cuenta a un Cliente
- El Mesero recoge el Efectivo de un Cliente
- El Mesero recoge la TarjetaDeCredito de un Cliente

Esto, sin duda, desordenará al modelo y lo hará difícil de comprender. Una forma más eficiente es examinar tales asociaciones, utilizar la menor cantidad de etiquetas y adjuntar las clases de asociación adecuadas.

www.elsolucionario.net

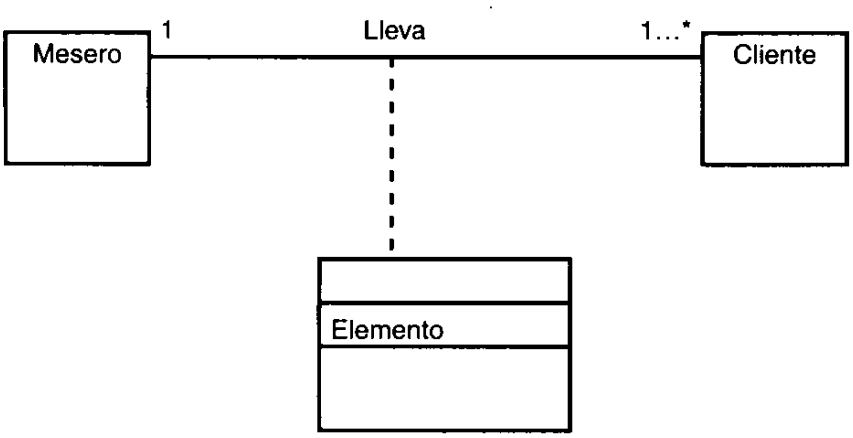
www.elsolucionario.net

El trabajo de un Mesero es, aparentemente, el de traer y llevar cosas. "Recoger" se sobreentiende como "llevar" y "servir" como "traer". Etiquetaremos esta asociación de la clase Mesero como "llevar" y "traer". Adjuntaremos una clase de asociación, y en ella indicaremos lo que se lleva o se trae. Para ello, le daremos un atributo llamado *elemento* y le asignaremos un tipo numérico. Los valores posibles del atributo son los diversos elementos que el Mesero podría llevar o traer.

La figura 17.7 le muestra lo anterior una vez puesto en efecto.

**FIGURA 17.7**

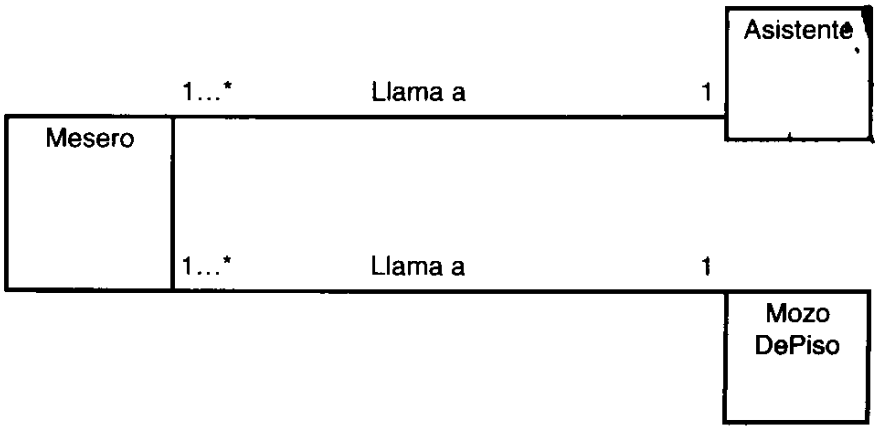
*El uso de clases de asociación en las asociaciones del Mesero.*




El Mesero también se asocia con un Asistente y un Mozo de piso, como muestra la figura 17.8.

**FIGURA 17.8**

*Otras asociaciones con el Mesero.*




## Asociaciones con el Chef

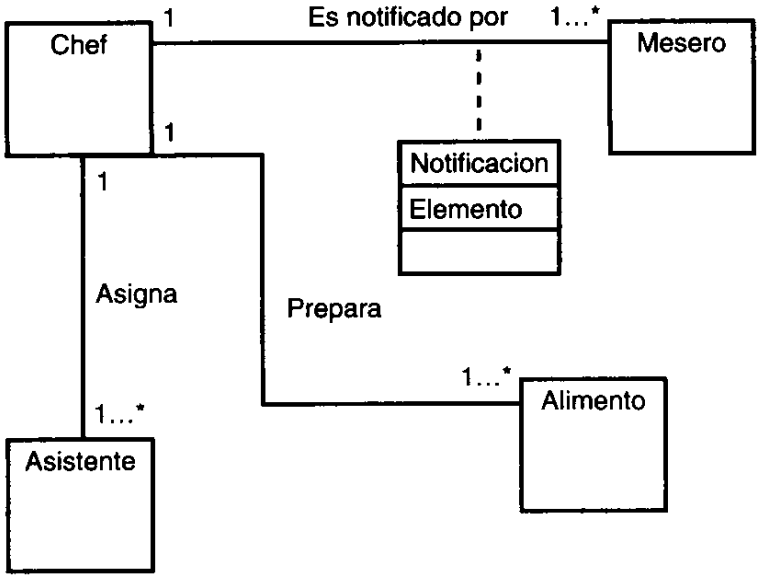
El Chef se asocia con el Asistente, el Mesero y con el Alimento, como en la figura 17.9.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 17.9**

*Asociaciones con el Chef.*

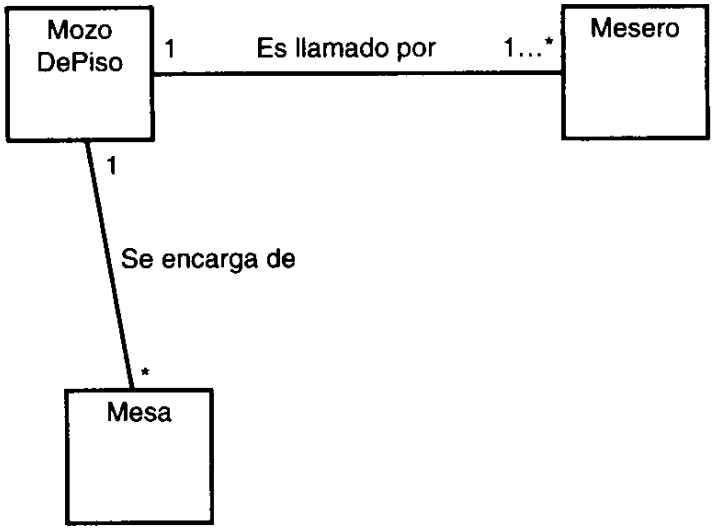



## Asociaciones con el Mozo de piso

El Mozo de piso realiza algunas tareas, como se establece en la figura 17.10.

**FIGURA 17.10**

*Asociaciones con el Mozo de piso.*




## Asociaciones con el Gerente

El Gerente es una clase nueva que hemos derivado a partir del análisis del dominio. Esta clase se asocia con muchas otras hacia las cuales hemos desarrollado las siguientes frases:

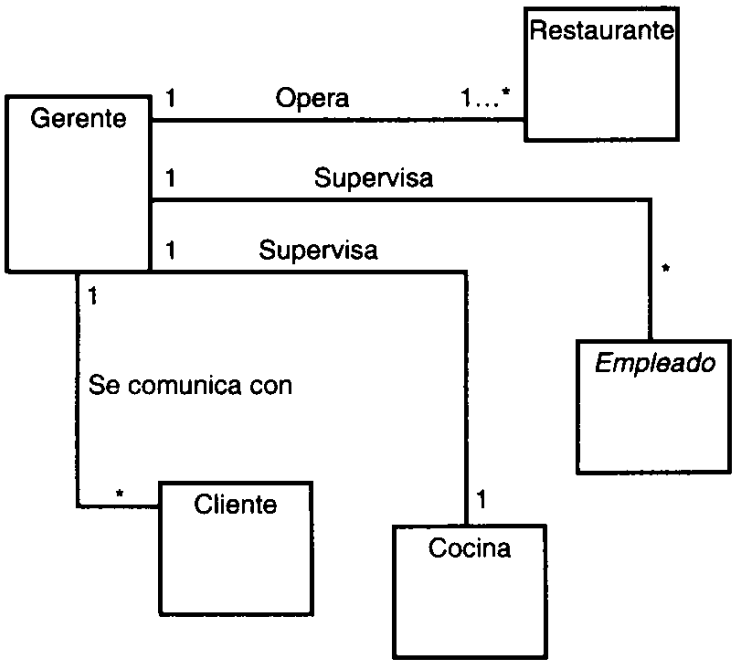
- El Gerente opera el Restaurante
- El Gerente supervisa a los Empleados
- El Gerente supervisa la Cocina
- El Gerente se comunica con el Cliente

La figura 17.11 modela tales asociaciones.

www.elsolucionario.net

**FIGURA 17.11**

*Asociaciones con el Gerente.*




## Una digresión

Algo que podría imaginar es que debiera eliminar los sustantivos que son roles en las asociaciones y contar sólo con una clase genérica, como Empleado. En la asociación, pondría el nombre del rol cerca de la línea de vida adecuada de la asociación.

En algunos contextos (como un sistema de nóminas), ello funcionaría bien. En éste tal vez no. Analice las siguientes asociaciones:

- El Mesero trae al Cliente
- El Mesero lleva del Cliente
- El Mesero trae al Chef
- El Mesero lleva del Chef
- El Mesero llama al Mozo de piso

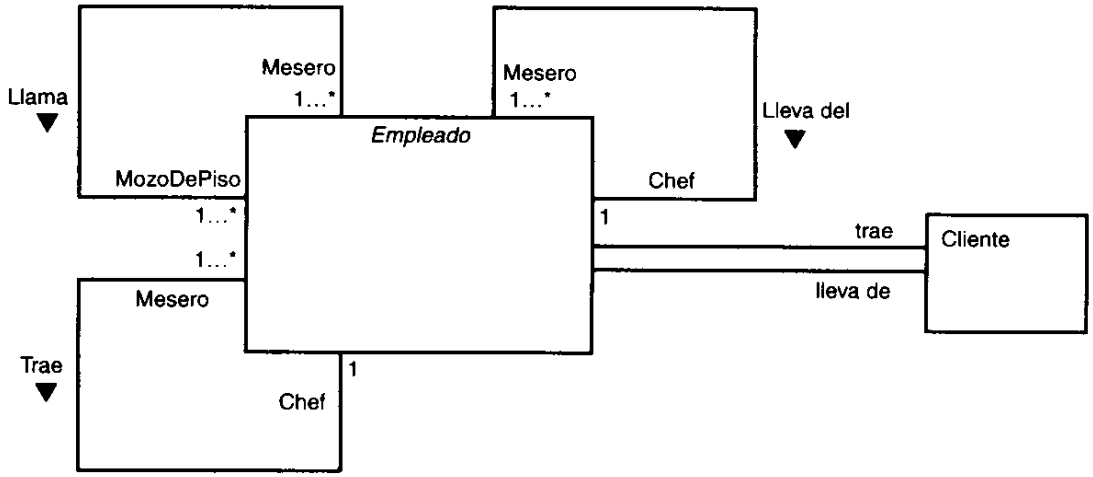
El diagrama luciría como en la figura 17.12.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 17.12**

*Modelado con la clase Empleado.*




Como puede advertir, los iconos de clase del diagrama se hacen densos y poco claros, y aún no hemos incluido las clases de asociación.

En todo lo que se relacione con el modelado, deje que le guíe la comprensión.

# Formación de agregados y objetos compuestos

Ya hemos conformado y bautizado clases abstractas y asociaciones, pero hay otra dimensión organizacional. El siguiente paso es localizar clases que sean componentes de otras. En este dominio ello no deberá ser difícil. Por ejemplo, un Alimento consta de un Entremés, un PlatoFuerte, una Bebida y un Postre. El Entremés y el Postre son opcionales. A su vez, los componentes se encuentran en un orden específico que deseamos conservar en nuestro modelo.

He aquí algunas otras composiciones:

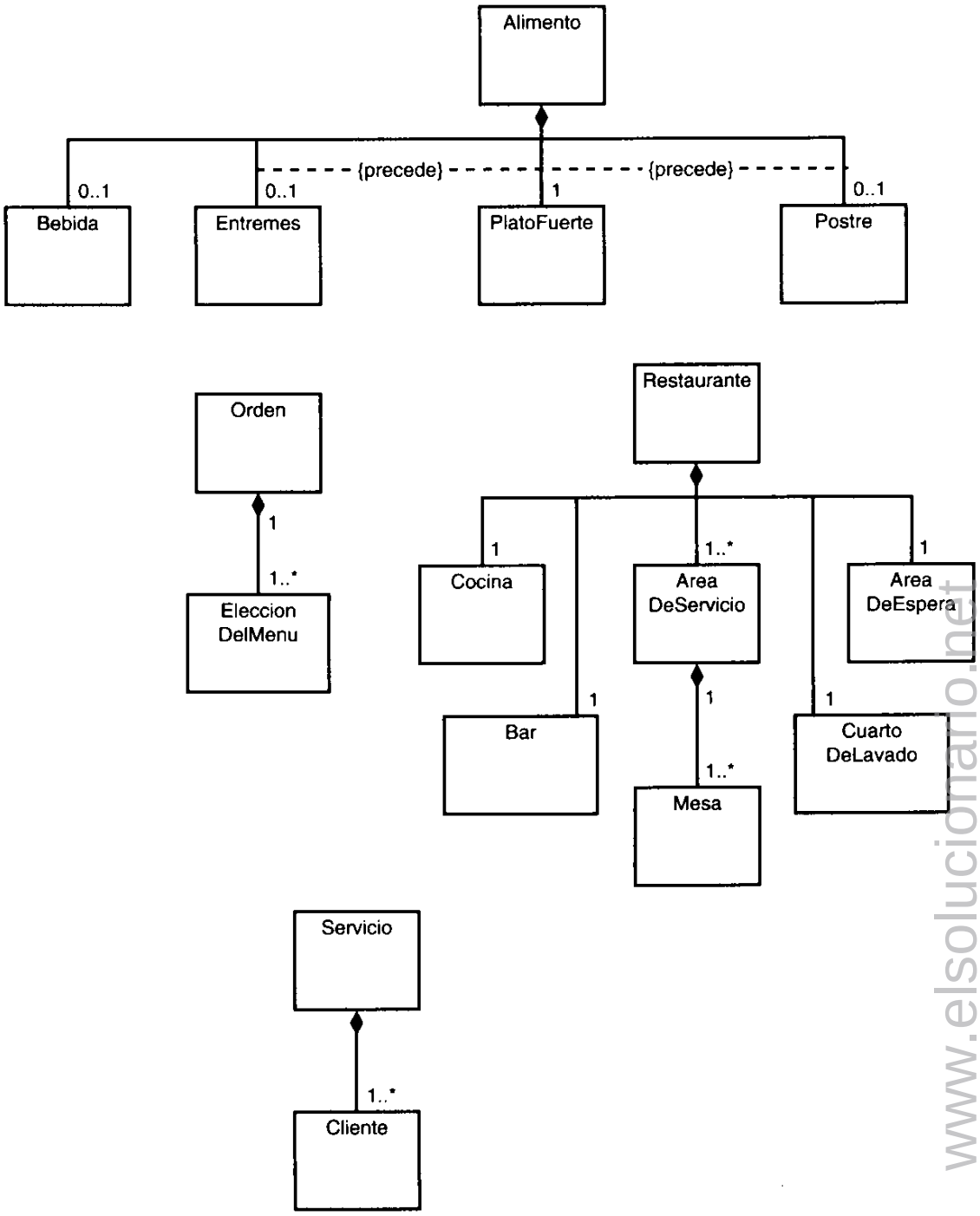
- Una Orden consta de una o varias EleccionesDeMenu
- Un Restaurante consta de una Cocina, una o varias AreasDeServicio, un AreaDeEspera, un Bar y un CuartoDeLavado
- Un AreaDeServicio consta de una o varias Mesas
- Un Servicio consta de uno o varios Clientes

En cada caso, el componente es miembro exclusivo de un agregado, de modo que la figura 17.13 los modela a todos ellos como objetos compuestos.

www.elsolucionario.net

**FIGURA 17.13**

*Objetos compuestos en el dominio Restaurante.*




# Llenado de las clases

Las entrevistas y sesiones consecuentes serán muy útiles para dar cuerpo a nuestras clases. Tenga en cuenta que a partir de este momento, un modelador de objetos deberá estar en cada sesión y depurará el modelo al mismo tiempo. Podemos comenzar con la depuración mediante la adición de algunos atributos y operaciones.

Nuestras clases más importantes parecen ser las de Cliente, Mesero, Chef, Gerente y Asistente. Verifique si hay alguna otra clase importante.

www.elsolucionario.net

www.elsolucionario.net

## El Cliente

¿Cuáles son los atributos obvios para un Cliente? He aquí algunos:

- nombre
- horaLlegada
- orden
- horaServicio

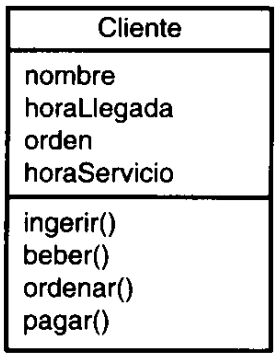
¿Qué hay con las operaciones? Nuestra lista de verbos nos podría servir de guía (aunque no nos deberá limitar). Algunas operaciones del Cliente son:

- ingerir()
- beber()
- estarFeliz (¡bromeaba!)
- ordenar()
- pagar()

La figura 17.14 le muestra la clase Cliente.

**FIGURA 17.14**

*La clase Cliente.*




## El Empleado

El Mesero, Chef, Gerente y Asistente son todas clases secundarias de la clase abstracta Empleado. Por ello, asignaremos atributos a Empleado y las clases secundarias los heredarán. Algunos de ellos son:

- nombre
- domicilio
- numeroSeguroSocial
- aniosExperiencia
- fechaContratacion
- salario

www.elsolucionario.net

www.elsolucionario.net

Para el asistente, hay cosas que son un poco más complejas. Primero, necesitaremos un atributo llamado *trabajaCon* dado que un Asistente podría ayudar a un Mesero o a un Chef. Este atributo será de tipo numérico.

Las operaciones serán específicas para cada clase secundaria. Para el Mesero, tales operaciones parecen ser adecuadas y pueden verse en la figura 17.15:

- llevar()
- servir()
- recoger()
- llamar()
- verificarEstadoDeLaOrden()

Para el Chef:

- preparar()
- cocinar()
- darPrioridad()
- crearReceta()

Para el Asistente:

- preparar()
- cocinar()
- servirPan()
- servirAgua()

Las operaciones del Gerente serían:

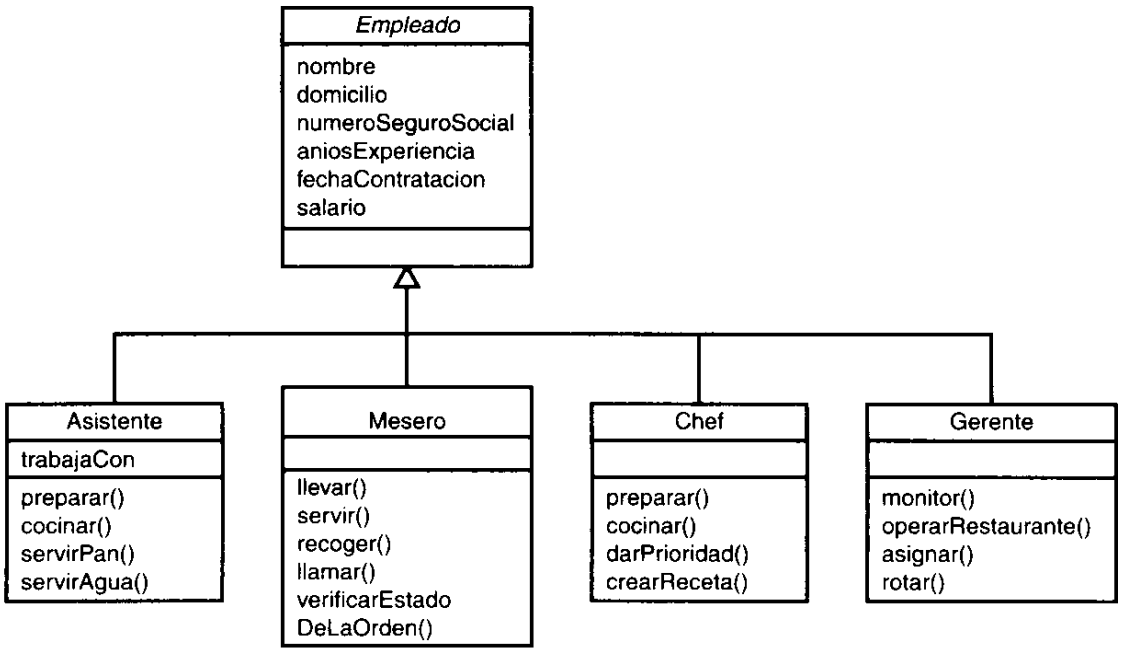
- supervisar()
- operarRestaurante()
- asignar()
- rotar()

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 17.15**

*La clase Empleado y sus clases secundarias.*




## La Cuenta

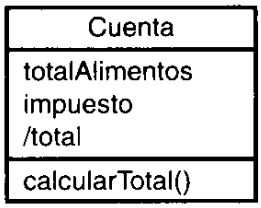
La Cuenta es, obviamente, una clase importante pues contiene la información del dinero que costó la comida. Sus atributos son:

- totalAlimentos
- impuesto
- total

Dado que total es la suma de totalAlimentos e impuesto, es una variable derivada. Nuestro modelo (vea la figura 17.16) lo refleja. La operación de la Cuenta es calcularTotal(totalAlimentos, impuesto).

**FIGURA 17.16**

*La clase Cuenta.*




# Detalles generales de los modelos

Hasta este punto ya ha recopilado bastante información. He aquí algunas sugerencias para ayudarle a mantenerlo todo organizado.

## Diccionario del modelo

Cuando conjugue los resultados de las entrevistas, procesos del negocio y análisis de dominio, mantenga un diccionario del modelo. Éste es un glosario de terminología en el modelo. Le ayudará a mantener la consistencia y a evitar la ambigüedad.

www.elsolucionario.net

www.elsolucionario.net

Por ejemplo: en nuestro dominio del restaurante, el término "menú" es prominente. Este término significa una cosa para un restaurantero y otra para un desarrollador de interfaces gráficas. "Servidor", el término propuesto pero rechazado, podría tener implicaciones similares: mientras el restaurantero podría pensar en un "mesero", un ingeniero de sistemas podría pensar en otra cosa completamente distinta. Si tiene definiciones con las que todos estén de acuerdo, o si está al menos consciente de qué palabras podrían causar una gran confusión, evitará muchos problemas subsecuentes. La mayoría de las herramientas de modelado le permiten generar un diccionario conforme cree el modelo.

## Organización del diagrama

Otra sugerencia tiene que ver con la organización del diagrama. No es recomendable tener todos los detalles de su modelo de clases en un enorme diagrama. Necesitará un diagrama principal que muestre todas las conexiones, asociaciones y generalizaciones, pero será mejor omitir los atributos y operaciones de él. Podrá enfocarse en determinadas clases si las coloca en diagramas por separado. Normalmente, las herramientas de modelado le permiten organizar sus diagramas mediante una vinculación adecuada entre ellas.

# Lecciones aprendidas

¿Qué es lo que hemos aprendido en nuestro análisis de dominio?

- La entrevista del proceso del negocio otorga las bases para el análisis del dominio
- Los sustantivos resultantes pueden convertirse en candidatos para clases
- Hay que eliminar los sustantivos que son atributos, y los que son sinónimos de otros sustantivos de la lista, así como los que representen a clases fuera del ámbito del dominio
- No pierda de vista la oportunidad de agregar clases que podrían no haberse mencionado durante la entrevista del proceso del negocio
- Válgase de algunos de los verbos o construcciones verbales como etiquetas para las asociaciones
- Agrupe las clases y utilice los nombres de los agrupamientos como clases abstractas
- Agrupe clases en agregados u objetos compuestos
- Cambie el nombre de las clases para mayor legibilidad
- Algunas asociaciones pueden ser tripartitas (esto es, que involucran a tres clases)
- Válgase del sentido común para denominar asociaciones y establecer multiplicidades

En la siguiente hora nos iremos del entorno conceptual a los detalles relacionados con el sistema.

www.elsolucionario.net

www.elsolucionario.net

# Resumen

Esta hora continuó con el análisis conceptual que inició en la hora anterior. La entrevista del proceso del negocio da por resultado los fundamentos para el análisis del dominio. Los sustantivos, verbos y construcciones verbales de la entrevista son los candidatos para el diagrama de clases inicial que definen al dominio Restaurante. El sentido común le indicará cuáles utilizar y cuáles eliminar. Es posible que agregue clases conforme haga el análisis.

El modelador de objetos agregará sustancia a este diagrama mediante la derivación de clases abstractas, asociaciones y multiplicidades. La derivación de agregados u objetos compuestos le ayudará a organizar al modelo. Serán necesarias otras entrevistas y sesiones para dar cuerpo completamente en el modelo, aunque es posible empezar a agregar atributos y operaciones en este punto.

# Preguntas y respuestas

**P ¿Cómo sabré cuáles clases eliminar de la lista de clases candidatas?**

**R** Mediante el sentido común, elimine los nombres de clases redundantes y esté consciente de los nombres que sean atributos. Elimine los nombres de las clases que estén fuera del ámbito del dominio que está analizando. Recuerde que también podrá agregar clases.

# Taller

Este taller verifica la muy importante aptitud para el análisis del dominio, requerida para la creación y desarrollo de un diagrama de clases. Las respuestas se encuentran en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿De qué forma utilizaremos los sustantivos obtenidos en la entrevista con un experto?
2. ¿Y de qué forma utilizaremos los verbos y construcciones verbales?
3. ¿Qué es una asociación "tripartita"?
4. ¿Cómo modelaría una asociación tripartita?

www.elsolucionario.net

www.elsolucionario.net

## Ejercicios

1. Revise las asociaciones tripartitas del cliente con el EncargadoDelGuardarropa. Utilice una clase de asociación para modelarlas de forma eficiente.
2. Si ha seguido con atención la entrevista y el análisis del dominio, tal vez habrá pensado en algunas clases que no aparecieron en ninguna de tales instancias. Una de ellas sería el Cajero. Conforme una asociación entre el Mesero y el Cajero. Utilice una clase de asociación en caso de ser necesario. Si puede pensar en otras clases, incorpórelas en el análisis del dominio.
3. Nuestro objeto compuesto Restaurante incluye sólo a clases "físicas": áreas como la Cocina y Bar. Tal vez podría decirme que un restaurante también incluye a gente. Revise el objeto compuesto Restaurante e incluya a los empleados en el diagrama. ¿Al incluir a los empleados se convierte al objeto compuesto en un agregado?
4. Además de los atributos y operaciones, apunté en la hora 3, "Uso de la orientación a objetos", que podría representar la responsabilidad de una clase. En la clase Mesero, agregue un panel de responsabilidades y asígnele una descripción de la responsabilidad propia del mesero.
5. Continúe con el dominio Biblioteca de los ejercicios de la hora 16, "Presentación del caso por estudiar", y desarrolle un diagrama de clases.

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 18

# Recopilación de las necesidades del sistema

Las dos horas anteriores trataron de cuestiones conceptuales respecto al dominio. Derivó procesos del negocio y generó diagramas de clases. Ahora iremos al sistema.

En esta hora se tratarán los siguientes temas:

- Cómo vislumbrar el sistema
- La sesión Desarrollo conjunto de aplicaciones (JAD)
- Cómo organizar las necesidades del sistema
- El aprovechamiento de los casos de uso

Los señores LaHudra, Nar y Goniff están impresionados. Han visto el resultado de su equipo de desarrollo y saben que el proyecto va en la dirección correcta. Todos parecen tener una buena comprensión del dominio Restaurante, tanto que los restauranteros de la División de Restaurantes de LNG dicen que los diagramas han aclarado su propio concepto de las operaciones de un restaurante.

www.elsolucionario.net

www.elsolucionario.net

Ahora el equipo tendrá que trabajar en la parte medular, es decir, en la parte técnica para el restaurante del futuro. Ya cuentan con los procesos del negocio y los diagramas de clases. Ya pueden empezar a codificar ¿cierto? ¡No! Ni siquiera están cerca de escribir un programa. Antes que nada, tienen que desarrollar una visión del sistema.

La mayoría de los proyectos empiezan con frases como "Generar una base de datos con la información del cliente y hacerla fácil de usar para que los empleados puedan utilizarla con un mínimo de capacitación" o "Crear un entorno de ayuda computarizado que resuelva los problemas en menos de un minuto". Aquí, el equipo de desarrollo ha comenzado la misión de "usar la tecnología para construir el restaurante del futuro". Tienen que vislumbrar a este restaurante basado en tecnología para tener una idea de la forma en que el personal del restaurante trabajará en él. Trabajan en un nivel en el que un equipo de desarrollo usualmente no llega, y LaHudra, Nar y Goniff confían en ellos.

El equipo utilizará el conocimiento del proceso del negocio y el recién adquirido conocimiento del dominio para ver en qué lugar la tecnología podría mejorar la experiencia de comer en un restaurante. Escuchemos una reunión del equipo. Los integrantes son un analista, un modelador, un restaurantero, un mesero, un chef y un ingeniero de sistemas. Habrá un moderador en la reunión.

# Desarrollo de la idea

Moderador: "Al analizar nuestros diagramas de procesos del negocio, pienso que podemos ver varios lugares donde podría ser útil la tecnología basada en computadoras. Haré una lista en la pizarra. ¿Quién quiere empezar?"

El moderador distribuye copias de la figura 18.1, el diagrama de procesos del negocio para "Servir a un cliente", y la figura 18.2, el de "Preparar un platillo".

Analista: "Sí, aparentemente el negocio del restaurante, como casi cualquier otro, depende de la transmisión de la información. Si podemos agilizar tal transmisión (algo en lo que la tecnología se especializa) cumpliremos nuestra meta."

Restaurantero: "No me queda muy claro. ¿Qué quieren decir con 'transmisión de la información'? Siempre pensé que mi negocio se centraba en el movimiento de alimentos."

www.elsolucionario.net

www.elsolucionario.net

Ingeniero de sistemas: "Yo le puedo explicar. Cuando el cliente hace una orden, le da información al mesero... por cierto, recordemos que mesero es alguien que sirve las mesas. Una vez que el mesero tiene la información, se la lleva al chef, con ello está transmitiendo información."

Moderador: "¿Dónde más se transmite información en el diagrama?"

Mesero: "Creo que cuando un cliente me pide que le informe cómo va el avance de su orden y yo le pregunto al chef, ello también puede traducirse como transmisión de información ¿no?"

Analista: "Así es."

Chef: "Transmisión, tecnicismos. No se ofendan, pero yo no me emociono para nada cuando un mesero entra y me pregunta cuánto más tardaré en terminar de preparar un platillo. Tardará lo que tenga que tardarse y no puedo ser molestado."

Moderador: "Bueno, tal vez podríamos hacer algo por reducir su molestia. ¿Ven algunos otros puntos de transmisión de información?"

En este caso el moderador intenta suavizar las cosas con el chef para que éste continúe involucrado.

Restaurantero: "¿Qué tal cuando el mesero hace la sugerencia del día? ¿O cuando responde a una pregunta acerca del menú?"

Moderador: "¡Por supuesto!"

Chef: "En ocasiones yo también tengo que dar respuestas. Las personas me envían al mesero para pedirme alguna receta en particular. A veces se las envío mediante el mesero y, si no hay mucho trabajo, yo mismo voy y hablo con el cliente. A ellos les gusta eso."

Mesero: "Pues hay cierta información que no me gusta transmitir. Un cliente hace una orden, voy y se la paso al chef, y luego de quince minutos, cuando regreso a la cocina por cualquier otra cosa, oigo que ya se acabaron los ingredientes para tal orden. Yo tengo que ir con el cliente y sugerirle que ordene alguna otra cosa. Generalmente eso molesta mucho al cliente y a mí también, debido a que reduce mi propina."

Analista: "Tal vez podríamos agregar esto al proceso del negocio..."

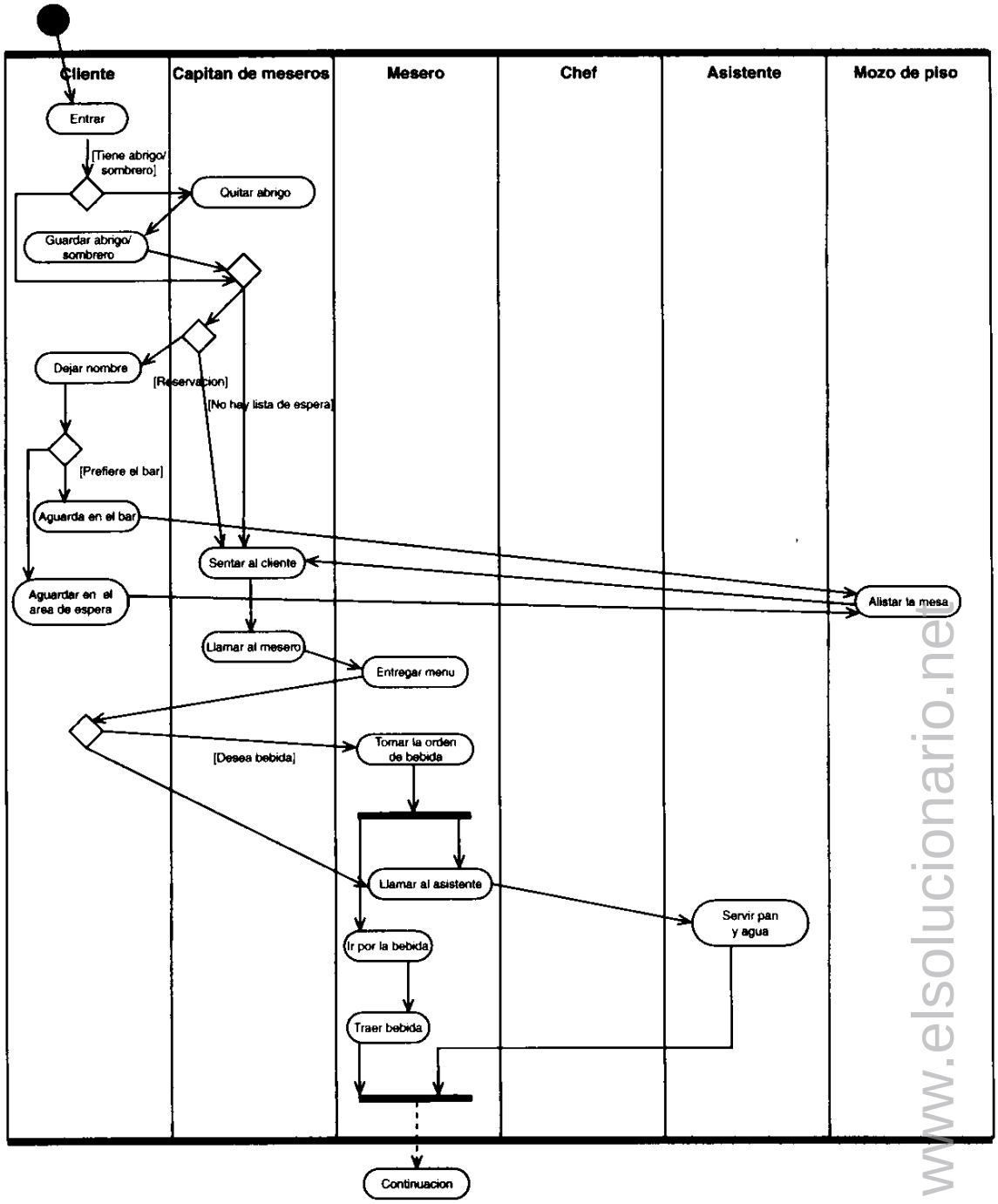
Moderador: "Tal vez. Pienso que, si están de acuerdo, podríamos tratarlo en otra reunión."

www.elsolucionario.net

www.elsolucionario.net

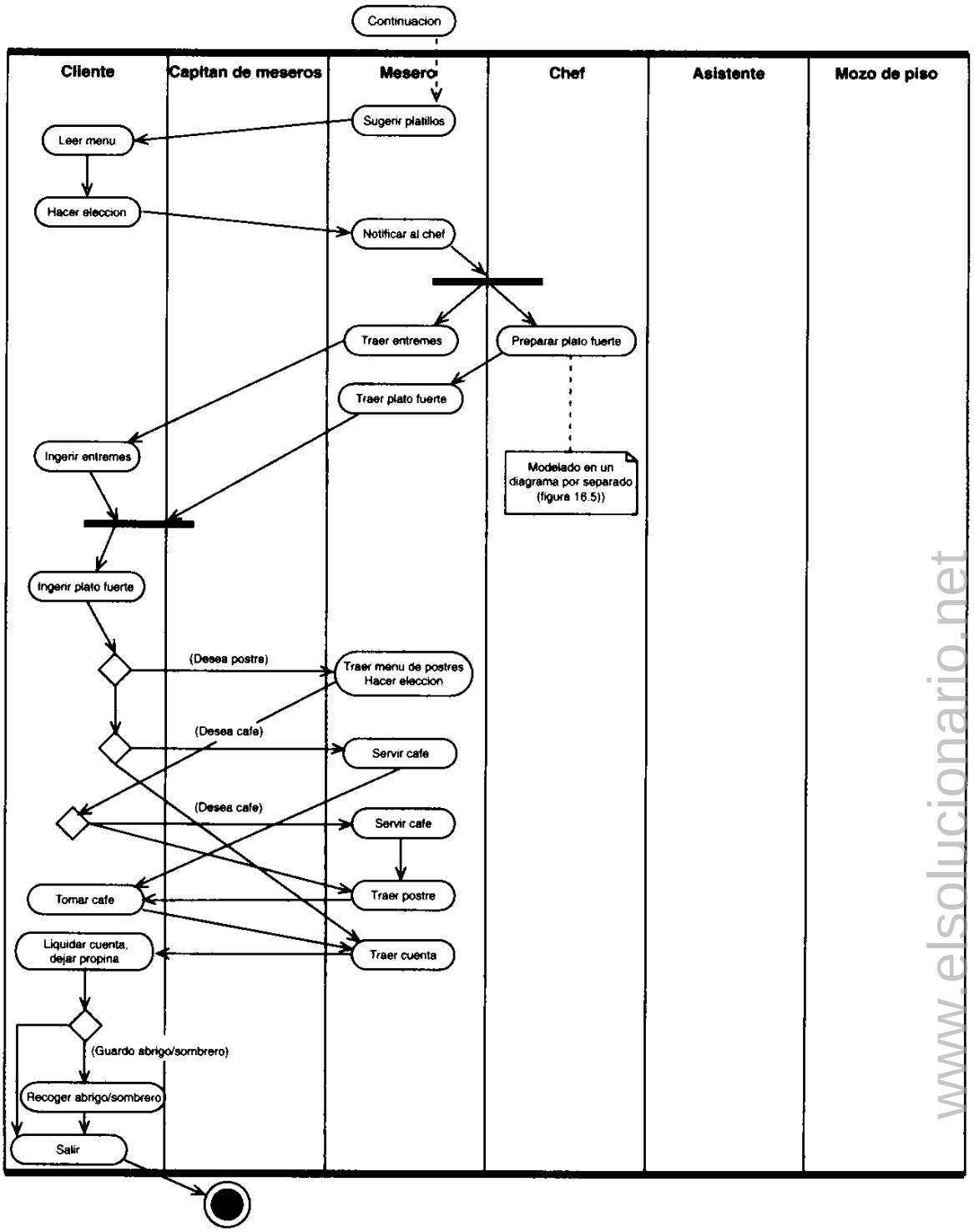
**FIGURA 18.1**

*Un diagrama de procesos del negocio para "Servir a un cliente".*




www.elsolucionario.net

www.elsolucionario.net

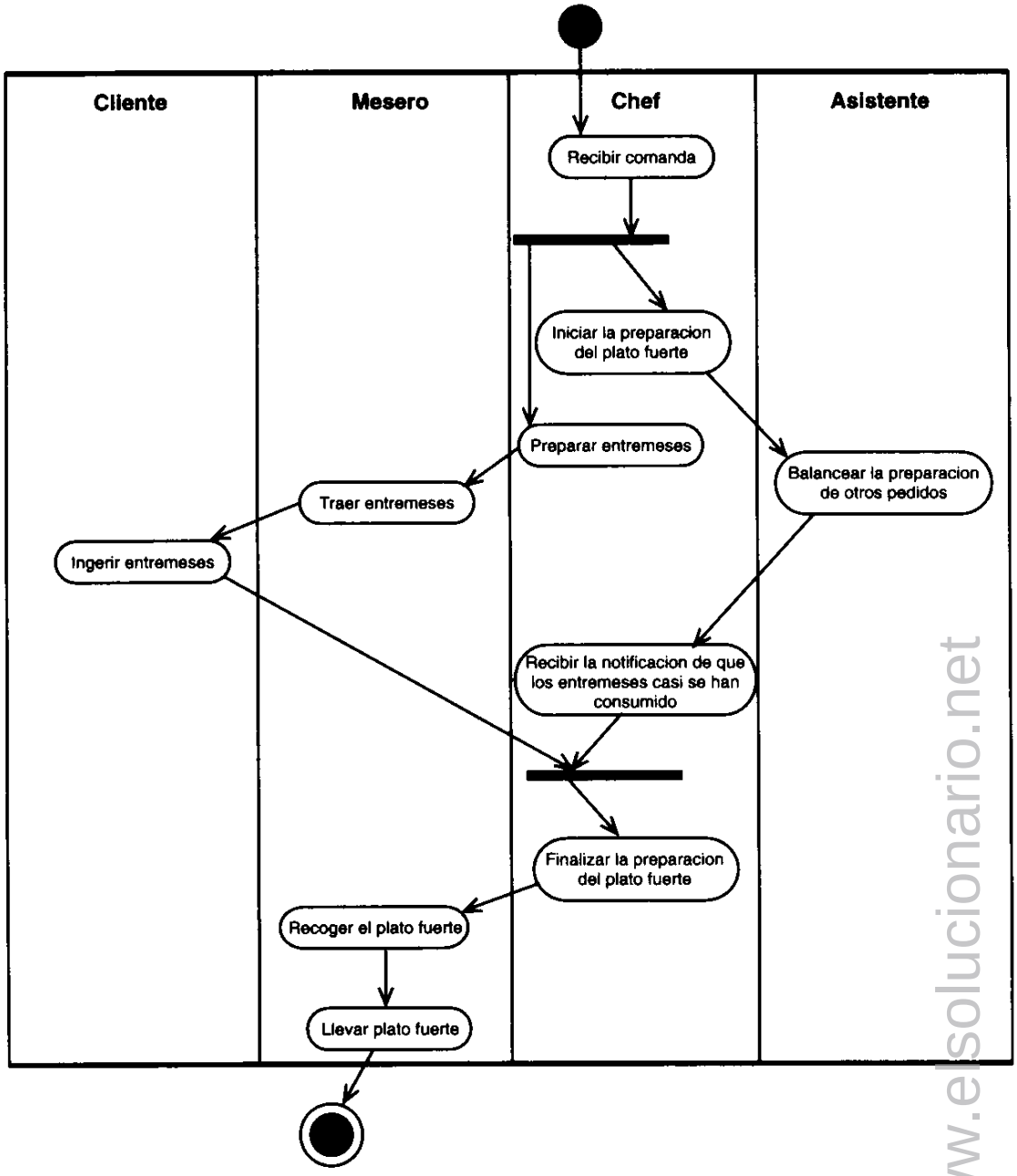



www.elsolucionario.net

www.elsolucionario.net

**FIGURA 18.2**

*Un diagrama de procesos del negocio para "Preparación de platillos".*



El moderador intenta mantener bien enfocada la reunión. Observe que el moderador evita utilizar la enojosa construcción "Sí, pero...".

Analista: "Sí, claro. No sería bueno desviarnos del tema."

Moderador: "Veamos dónde hemos llegado. De acuerdo con la lista que he hecho, la transmisión de la información se realiza cuando

- El cliente hace una orden
- El mesero lleva la comanda al chef
- El cliente pide al mesero el grado de avance de su orden
- El mesero da la sugerencia del día

www.elsolucionario.net

www.elsolucionario.net

- El mesero responde a una pregunta de algo en el menú
- El chef responde a inquietudes respecto a alguna receta."

Es muy adecuado que, en la entrevista del proceso del negocio, el moderador haga una pausa para resumir.

Analista: "Sé que no está en ninguno de nuestros diagramas de negocios, pero ¿qué acaso el cliente no tiene preguntas sobre algo en la cuenta? Cuando el mesero le da la respuesta, también se tiene transmisión de información."

Moderador: "Correcto. ¿Algo más de los procesos del negocio?"

Ingeniero de sistemas: "Creo que sí. ¿Qué hay de la famosa coordinación entre el mesero y el chef? Es decir, ¿en qué momento saben que debe servirse el plato fuerte una vez que los clientes han terminado sus entremeses? También, esto es información que se transmite."

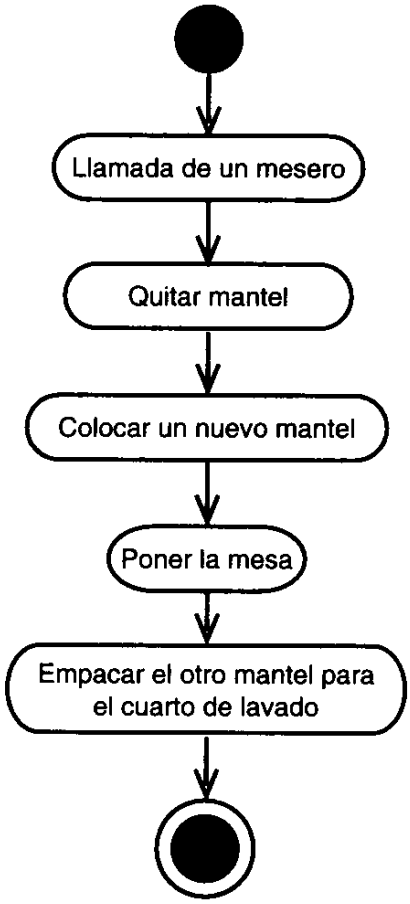
Analista: "De acuerdo. La información se transmite de distintas formas en este caso."

Restaurantero: "Sólo nos dio dos diagramas de procesos del negocio y recuerdo que creamos otro."

Moderador: "Así es. Aquí está el de 'Limpieza de una mesa'" (vea la figura 18.3).

**FIGURA 18.3**

*Un diagrama de procesos del negocio para "Limpieza de una mesa".*




www.elsolucionario.net

www.elsolucionario.net

Analista: "Parece que aquí sólo hay un caso de transmisión de información , pero apuesto que es muy importante: El mesero llama al mozo de piso para hacerle saber que es hora de limpiar la mesa."

Restaurantero: "Esto es muy importante. No puede sentar a un nuevo grupo de personas en la mesa si no está lista. Si la limpieza no se realiza tan rápido como sea posible, tendremos muchos clientes hambrientos, y molestos, amontonados en el bar y en el área de espera."

Modelador: "He ido modificando mis diagramas de clases mientras transcurre la reunión. ¿Puedo hacer una pregunta? ¿Sería bueno que nuestro sistema (como sea que luzca) nos permitiera evaluar nuestra eficiencia en conjunto para atender a nuestra clientela?"

Restaurantero: "¡Claro! De esa forma sabríamos dónde y cómo mejorar. ¿En qué está pensando?"

Modelador: "En nuestra clase Cliente hay un atributo horaLlegada y otro horaServicio. Quisiera agregar un atributo derivado que se llame tiempoEspera, que podría ser la diferencia entre horaLlegada y horaServicio. ¿Qué opinan?"

Restaurantero: "Me parece bien. Así sabremos qué tal estamos en cuestiones de eficiencia."

Analista: "Así es. Tendrán muchos datos con los que podrán jugar, por ejemplo, tiempoEspera como una función de la hora del día, o como una función de cuántos meseros estaban laborando en ese momento o cosas como esas..."

Modelador: "Hay otra posibilidad. ¿Y si agregamos otro atributo llamado horaSalida, y un atributo derivado llamado duracionComida que sería la diferencia entre horaServicio y horaSalida?"

Moderador: "Parece ser una buena sugerencia. La agregaré. ¿Alguna otra?"

Modelador: "Ya que estamos con los atributos de tiempo, ¿qué tal si agregamos algunos atributos de tiempo a las clases Mesero y Chef que indiquen al gerente el tiempo que se tardan en realizar el trabajo?"

Restaurantero: "Huy, no… La idea de supervisar el rendimiento de los empleados no es muy agradable para ellos y para mí tampoco. No porque sean perezosos (no lo son), sino porque no es agradable sentirse acosado por una especie de capataz. Es mejor que todos se sientan a gusto en el trabajo; de esta manera, el restaurante será mejor y los clientes también se sentirán más a gusto."

www.elsolucionario.net

www.elsolucionario.net

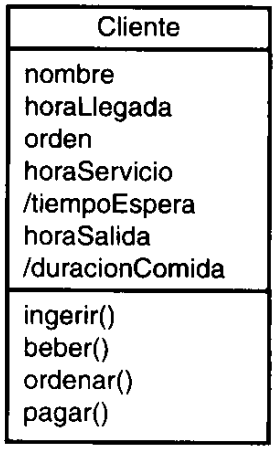
Chef: "Sí, estoy de acuerdo. Como lo dije, cuando se prepara un platillo se tardará lo que tenga que tardarse. No quiero tener un manojo de comandas y tener un capataz que me diga que tengo que tardarme cuatro minutos y medio menos en preparar una trucha a la almandina."

Mesero: "Y yo no quiero oír que me he tardado mucho en llevar el menú de postres cuando los clientes hayan terminado con sus alimentos. Hay muchas cosas involucradas."

Modelador: "Bien, bien, idea descartada. De hecho, ya que lo mencionan, creo que debo quitar 'supervisar' como operación de la clase Gerente. Entretanto, he aquí la forma en que luce la clase Cliente ahora." (Vea la figura 18.4.)

**FIGURA 18.4**

*La clase Cliente actualizada.*



Las ideas del modelador muestran que se encuentra modificando constantemente los diagramas de clases.

El debate entre el modelador, el restaurantero y el mesero dejan entrever un punto crucial: es necesario que las personas del negocio participen en el desarrollo del sistema. Sin las opiniones tanto del restaurantero como del mesero, el proyecto de desarrollo podría haber gastado tiempo y dinero en instaurar algunas características para la supervisión del rendimiento que no servirían a posteriori. Los empleados podrían haber reaccionado de manera negativa, lo que causaría repercusiones en el sistema y, eventualmente, en el restaurante.

Moderador: "Por lo que he escuchado, parece que podemos distinguir dos tipos de agilidad. Uno de ellos involucra la velocidad con que se transmite la información, y la otra a la rapidez con que cada empleado realiza una tarea. El sentir del grupo deja entrever que la segunda es una molestia, pero la primera no. ¿Estoy en lo correcto?"

(Todos asienten).

Analista: "Bien, ya que estamos de acuerdo en ello, ¿podríamos ver algunas otras ideas respecto a lo que el sistema haría específicamente?"

Moderador: "Claro. ¿Alguna idea?"

www.elsolucionario.net

www.elsolucionario.net

Mesero: "Cuando transmito toda esta información, camino mucho en el transcurso de una tarde. En ocasiones tengo que trabajar en un área que está lejos de la cocina. El caminar tanto se lleva tiempo —sin contar las suelas de mis zapatos—."

Analista: "Bien, pues parece que tendremos que hacer algo para eliminar, o al menos reducir, el tiempo muerto. Con ello podríamos agilizar la transmisión de la información."

Moderador: "¿Tiempo muerto?"

Analista: "Sí. Nuestro sistema debe evitar que los meseros caminen de más. Es obvio que tendrán que ir a la cocina para llevar la orden a los clientes, pero ¿y qué tal si logramos que sea a lo único que vayan a la cocina? ¿Y suponga que pudieran ir a ella justo a tiempo para traer la orden?"

Ingeniero de sistemas: "Pues creo que ya nos estamos concentrando en algo importante. Se me ocurre que podríamos establecer una pequeña red que conecte a los meseros con la cocina, incluso con los mozos de piso. Así la información se transmitiría con mucha rapidez."

Analista: "No quisiera sonar demasiado analítico aquí pero... ¿Una pequeña red? Tendrán que brincar cables para llegar a las terminales. En lugar de caminar a la cocina, tendrán que buscar una computadora. A mí me parece que es como matar moscas a cañonazos. ¿Qué nos ahorraría?"

Ingeniero de sistemas: "Tal vez nada, como lo has descrito. Incluso hasta el servicio podría empeorar. Pero así no era lo que yo pensaba."

Analista: "¿Entonces?"

Ingeniero de sistemas: "Bueno. Suponga que cada mesero y mozo de piso llevan consigo una terminal. Una pequeña, del tamaño de la palma de su mano. Y también supón que no usamos cables. Tendríamos una terminal de escritorio en la cocina y otra en la oficina del gerente."

Analista: "Ah, ya veo... El sistema del que hablas podría resolver muchas cosas. Por ejemplo, si el cliente hace sus órdenes, el mesero podría marcarlas en su computadora de mano y se irían a una terminal en la cocina. Ello eliminaría el paso, y los pasos, de caminar del área de servicio a la cocina."

Mesero: "Me gusta. ¿Qué tal si el cliente casi ha acabado sus entremeses y se lo indico a la cocina apretando algo en la computadora que dicen? Ello me evitará el tener que ir y decirle al chef que termine de preparar el plato principal."

Chef: "Y así veré el mensaje en la cocina. De hecho, todos mis asistentes podrían ver el mensaje al mismo tiempo, y podríamos tener los mensajes en una, dos o tres pantallas grandes. No tendré que ver cuál asistente estaba cocinando qué platillo ni decirles qué tanto deberían de haber avanzado. Ellos podrían tomar tal responsabilidad por sí mismos."

www.elsolucionario.net

www.elsolucionario.net

Ingeniero de sistemas: "Y cuando terminen la orden, podrían enviar un mensaje al mesero para hacérselo saber. No tendrá que estar regresando a la cocina para verificar."

Mesero: "Huy, pues suena muy bien. Incluso podría enviarle una señal a un mozo de piso para que limpie una mesa. No tendré que buscar a uno. Ello podría agilizar mucho las cosas."

Restaurantero: "¿Cómo podrían ustedes hacer esto?"

Ingeniero de sistemas: "No se preocupe de ello por el momento."

Moderador: "Bueno, entonces nuestro sistema será una red de área local inalámbrica con computadoras palmtop para los meseros y los mozos de piso, y con computadoras de escritorio en la cocina y en la oficina del gerente. Pero nos estamos olvidando de algo."

Analista: "¿De qué?"

Moderador: "De un nombre para el sistema."

Chef: "¿Qué tal 'EL EXPERTO CHEF'?"

Moderador: "¿Y tiene algún significado en particular? ¿Forman alguna sigla?"

Chef: "No. Sólo me gustó."

Analista: "¿Qué tal Red Interactiva Inalámbrica para Restaurantes? Podríamos llamarla RIIR."

Moderador: "Bueno... No lo sé."

Ingeniero de sistemas: "Bueno, tal vez podría ser Red Inalámbrica para la Comunicación Organizada. ¡Miren! Y ello deja las siglas RICO, que sería muy adecuado para el servicio en un restaurante."

Chef: "Me gusta."

Analista: "A mí también. El nombre deja entrever que el sistema nos enriquecería a nosotros, y tiene un significado muy directo."

Moderador: "Entonces, ¿Estamos de acuerdo con RICO? Bien, pienso que hemos terminado por el momento."

# Preparación para la recopilación de las necesidades

El equipo pasa los resultados de su reunión a los funcionarios corporativos. LaHudra no puede creer en su buena fortuna al explorar una nueva área. Nar está completamente abrumado. Goniff ve visiones de signos monetarios que bailan ante sus ojos. Los tres le dicen al equipo que prosiga.

www.elsolucionario.net

www.elsolucionario.net

Ahora que el equipo tiene una idea del sistema, ¿podrán iniciar su trabajo los programadores y los ingenieros de sistemas? Por supuesto que no. El equipo deberá adaptar el sistema RICO a las necesidades de los usuarios, no a la tecnología. Aunque ya cuentan con algunas ideas de la reunión del equipo, aún no han expuesto el concepto de RICO a un grupo de empleados y gerentes para obtener información e ideas desde su punto de vista como usuarios.

La siguiente acción GRAPPLE hace exactamente eso. En una sesión de Desarrollo conjunto de aplicaciones (JAD), el equipo obtendrá y documentará las necesidades del sistema. Con esto, podrán hacer algunas estimaciones de tiempo y dinero.

La sesión JAD se realiza en una sala de conferencias. Encabezada por un moderador, se denomina sesión "conjunta" dado que incluirá a miembros del equipo de desarrollo y a usuarios potenciales del sistema, así como a expertos del dominio. Los miembros del equipo de desarrollo en esta reunión son dos analistas que toman notas, un modelador, dos programadores y un ingeniero de sistemas. Los usuarios potenciales son tres meseros, dos chefs, dos restauranteros y dos mozos de piso.

El objetivo de esta reunión es producir un diagrama de paquetes que muestre los principales segmentos de funcionalidad del sistema. Cada paquete representará un segmento y contendrá casos de uso que detallarán de qué se trata cada segmento de funcionalidad.

Vamos a la sesión.

# La sesión JAD de necesidades

Moderador: "Para empezar, quiero agradecerles a todos haber venido a nuestra sesión. Estas sesiones nos pueden llevar algo de tiempo, pero también pueden ser muy divertidas. Lo que intentamos hacer es conocer las necesidades para un sistema llamado RICO: Red Inalámbrica para la Comunicación Organizada."

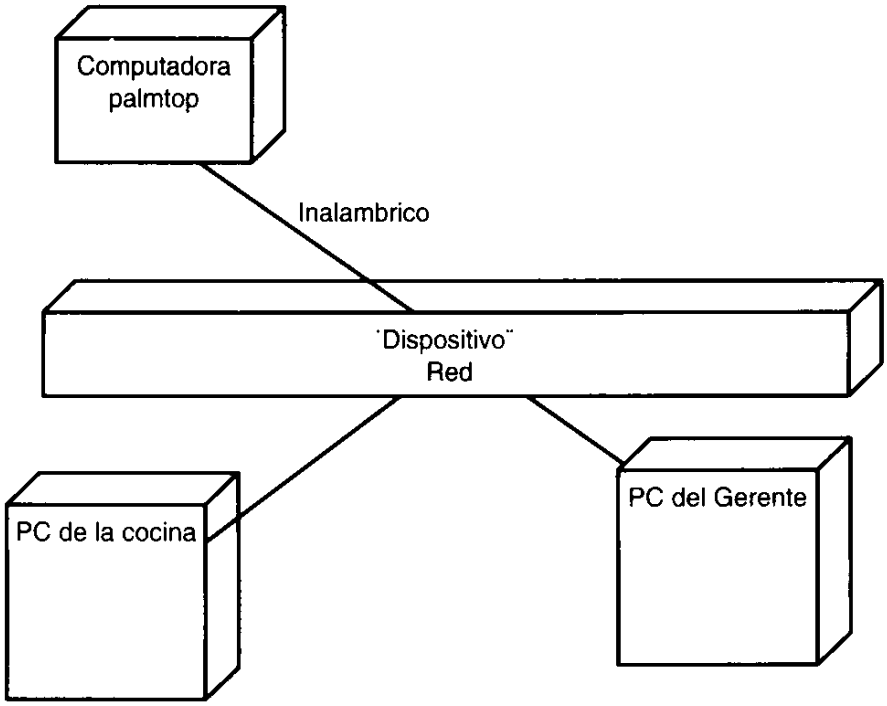
"El concepto de RICO es muy directo. La forma en que lo imaginamos es que los meseros tendrán en su poder una computadora palmtop y la utilizarán para comunicarse con la cocina y sus mozos de piso. Estos últimos también tendrán computadoras de este tipo y las utilizarán para comunicarse. La cocina tendrá una terminal de escritorio y una o varias pantallas. El gerente también tendrá una en su oficina. He aquí una figura de lo que estoy hablando." (Vea la figura 18.5.)

"Esperamos instalar a RICO en todos los restaurantes LNG, y queremos ayudarles a hacer su trabajo. Para lograrlo, necesitamos que nos digan qué es lo que desean que haga el sistema. Es decir, si el sistema ya estuviera en funcionamiento ¿qué querrían que hiciera?"

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 18.5**
*El sistema RICO.*




"Haremos esta pregunta una y otra vez. Al final de la sesión, tendremos un conjunto organizado de necesidades con las que todos estaremos de acuerdo. Piensen que es su carta a Santa Claus. Nos basaremos en esa carta para crear un proyecto que los programadores utilizarán para generar el sistema. Hay algo que quisiera que tuvieran en cuenta: necesitamos la comprensión e ideas de cada uno de ustedes, no importa cuál sea el nombre de su puesto."

Analista1: "¿Podríamos empezar por imaginarnos cuáles serían los principales segmentos de funcionalidad?"

Moderador: "Sí. ¿Están de acuerdo los miembros del grupo?"

Restaurantero2: "Si, mire... Yo no estuve en los debates anteriores, pero creo que esto es una buena idea. ¿Podríamos organizarlo de acuerdo a, por decir, las áreas del restaurante? Es decir, las áreas de servicio tienen ciertas necesidades, la cocina otras, el área de espera otras..."

Moderador: "Es una posibilidad."

Analista2: "Hum... Al ver los diagramas de procesos del negocio yo ya veo una organización."

Programador1: "¿Perdón?"

Analista2: "Por tarea. El chef tiene que realizar ciertas cosas, el mesero otras, y así."

Moderador: "Suena bien. ¿Estaríamos de acuerdo en organizarlo por tarea?"

(Todos asienten).

www.elsolucionario.net

www.elsolucionario.net

Moderador: "¡Bien! En los diagramas de procesos del negocio y de clases, las tareas que tenemos son Mesero, Chef, Mozo de piso, Asistente y Gerente."

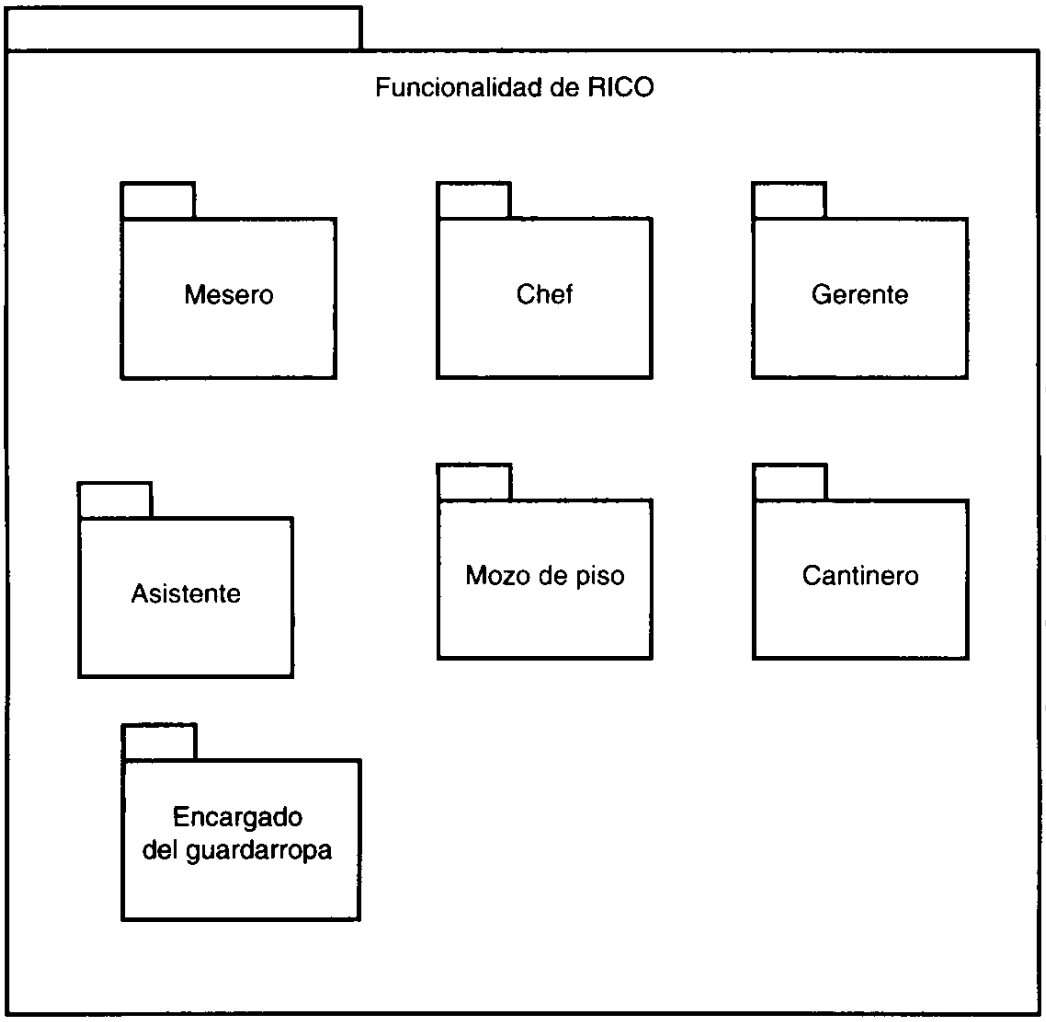
Restaurantero2: "¿No olvidaron algunas? No están el encargado del guardarropa y el cantinero."

Restaurantero1: "¡Vaya! ¿Cómo se me pudieron olvidar?"

Moderador: "Los agregaré a nuestra lista, y utilizaré los símbolos de paquetes del UML para estar al tanto." (Vea la figura 18.6.)

**FIGURA 18.6**

*Los paquetes de funcionalidad de RICO.*




Modelador: "Ya estoy en ello. Agregué información a nuestros diagramas de clases. Ya estaba la clase EncargadoDelGuardarropa. Lo que agregué fue la clase Cantinero."

Restaurantero2: "Me pregunto qué hace en su computadora laptop. ¿Me podría mostrar sus, hum, 'clases'?"

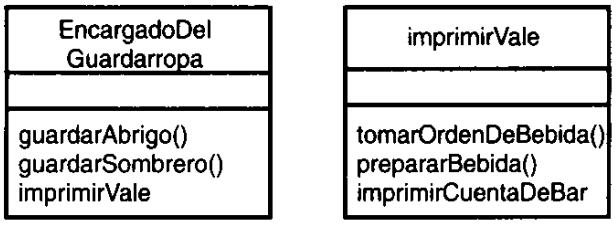
Modelador: "Sí, claro. Aquí están." (Vea la figura 18.7.)

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 18.7**

*Las clases EncargadoDelGuardarropa y Cantinero.*




Restaurantero2: "Muy interesante. Espero que en algún descanso se dé tiempo de explicarme lo que significa."

Moderador: "Ahora que contamos con las piezas principales, ¿alguien prefiere empezar con alguna tarea en particular?"

Mesero1: "¿Qué tal con la del Mesero?"

Moderador: "Me parece adecuado. Bien, ¿qué tipo de funcionalidad quisiera ver en este paquete? Recuerden, miembros del grupo, que aunque tratemos una tarea que no sea la de ustedes, pueden participar. Todas las opiniones son bienvenidas."

Mesero2: "Quisiera poder tomar una orden en mi pequeña computadora y enviarla a la cocina."

Moderador: "Muy bien. ¿Qué más?"

Mesero1: "Quisiera saber el grado de avance de una orden."

Chef2: "¿Puedo indicarle a un mesero cuando ya esté lista su orden?"

Moderador: "Sí y sí. Estos conceptos, como ven, los estoy escribiendo dentro de elipses rotuladas. Las llamaré 'casos de uso'. Les pediremos a algunos de ustedes que regresen y que nos ayuden a analizar tales casos de uso, pero ello será en otra reunión."

# El resultado

La sesión JAD continuó por el resto del día. Cuando los participantes terminaron, ya contaban con un conjunto de necesidades que aparecieron como casos de uso organizados en los paquetes.

Para el paquete mesero, los casos fueron:

- Tomar una orden
- Transmitir la orden a la cocina
- Cambiar una orden
- Recibir una notificación de la cocina

www.elsolucionario.net

www.elsolucionario.net

- Llevar un control del progreso de la orden
- Notificar al chef el progreso de los clientes en sus alimentos
- Totalizar una cuenta
- Imprimir una cuenta
- Llamar a un asistente
- Llamar a un mozo de piso
- Tomar una orden de bebida
- Transmitir una orden de bebida al bar
- Obtener un acuse de recibo
- Recibir una notificación del bar

Para el paquete chef, los casos de uso fueron:

- Almacenar una receta
- Recuperar una receta
- Notificar al mesero
- Recibir una petición del mesero
- Dar acuse de recibo a una petición del mesero
- Indicar el periodo de preparación
- Asignar una orden

Los casos de uso del mozo de piso fueron:

- Recibir una petición del mesero
- Dar acuse de recibo a una petición
- Indicar que una mesa ha sido limpiada

Los casos de uso para el asistente fueron:

- Recibir una petición del mesero
- Recibir una petición del chef
- Dar acuse de recibo de una petición
- Notificar que la petición se ha completado

Para el cantinero:

- Capturar una receta de bebida
- Obtener la receta de una bebida
- Recibir una notificación del mesero

www.elsolucionario.net

www.elsolucionario.net

- Recibir una petición del mesero
- Dar acuse de recibo de una petición
- Notificar que la petición se ha completado

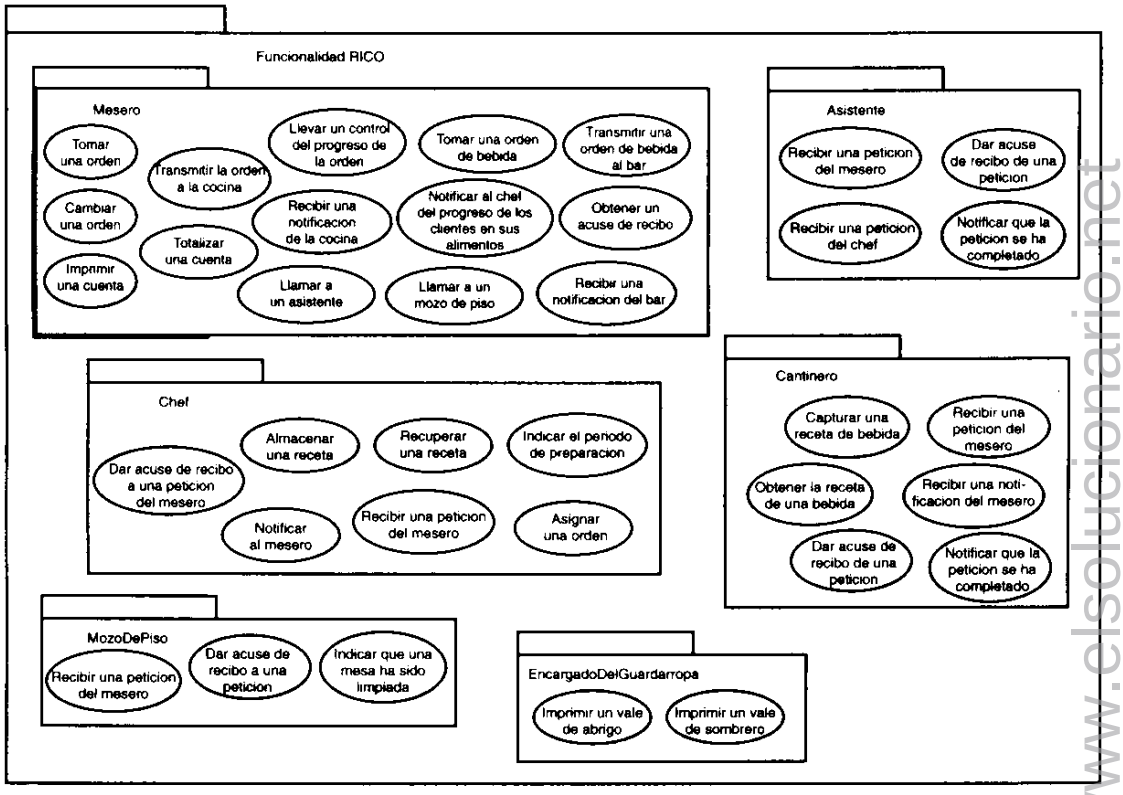
Y para el encargado del guardarropa:

- Imprimir un vale de abrigo
- Imprimir un vale de sombrero

La figura 18.8 muestra la forma en que todo lo anterior luce en el UML.

**FIGURA 18.8**

*El diagrama de paquetes de funcionalidad.*



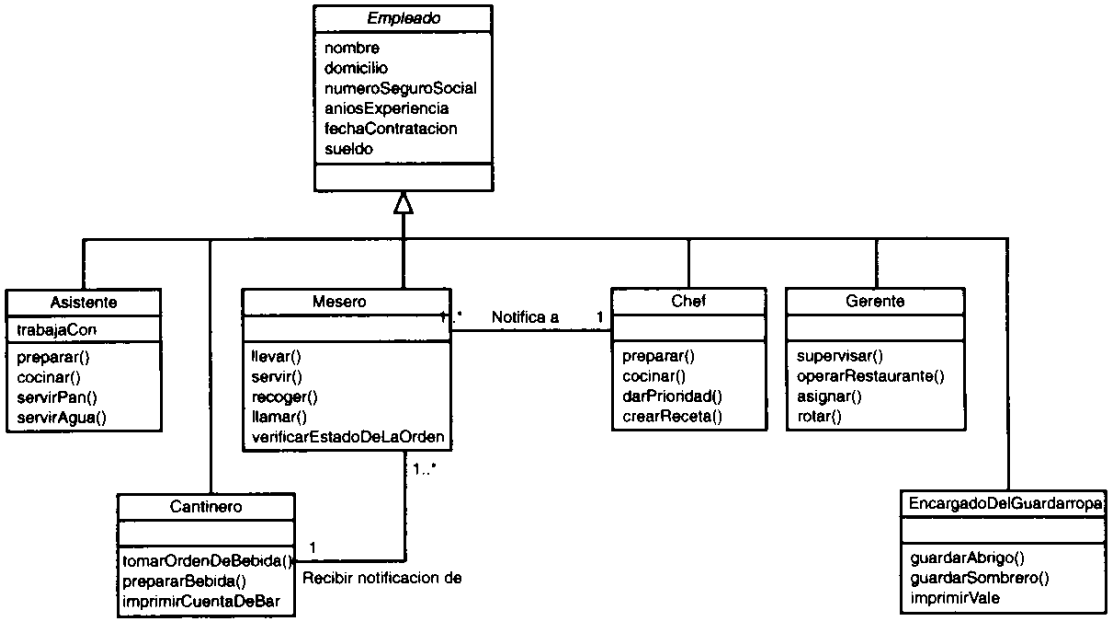

El modelador hará el desarrollo ulterior de los diagramas de clases mediante la adición de dos clases y asociaciones como en la figura 18.9.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 18.9**

*La información de las clases recién agregada.*




# ¿Ahora qué?

El documento de diseño que entregará el equipo a su cliente crece a pasos agigantados. Ahora incluye procesos del negocio, diagramas de clases y un conjunto de paquetes de funcionalidad.

¿Ahora empezará a codificar el equipo? En absoluto. En la siguiente hora empezarán a analizar el contenido de los paquetes.

# Resumen

En el contexto de la reunión, el equipo de desarrollo ha generado una visión del sistema de cómputo para el restaurante del futuro. Los miembros del equipo decidieron que la agilización en la transmisión de la información es la clave para el éxito del sistema, y traen a colación tipos de tecnología para lograrlo.

En una sesión JAD, el equipo de desarrollo se reúne con usuarios potenciales y expertos de dominio para obtener los requerimientos del sistema. El resultado es un diagrama de paquetes en donde cada paquete representa una sección principal de funcionalidad. Los casos de uso dentro de un paquete se basan en tal funcionalidad.

www.elsolucionario.net

www.elsolucionario.net

# Preguntas y respuestas

**P ¿Podría ser que algunos de los participantes de la sesión JAD sean los mismos que participaron en la reunión anterior en equipo?**

**R** Sí, y, de hecho, es recomendable. Tales personas podrían recordar detalles vitales que podrían no ser muy claros en las minutas.

**P Vi que los señores LaHudra, Nar y Goniff no participan en tales reuniones. ¿Podría alguien de tal nivel tomar parte en las reuniones y las sesiones JAD?**

**R** Estos en particular no. Sin embargo, en algunas empresas, los altos directivos podrían participar de manera activa al menos en parte de una sesión. Es difícil que un alto directivo tome parte en toda una sesión JAD.

**P ¿Siempre se dará el caso de que se tenga que organizar al sistema por tareas, como lo hicimos en este dominio?**

**R** No, no siempre. Aquí se hizo así porque fue conveniente para este dominio. De hecho, podríamos buscar una forma alternativa de hacerlo si nos concentramos en ello. Otros tipos de sistemas podrían requerir un trato distinto. Por ejemplo: un área de asistencia podría tener Recepción de llamadas, Resolución de problemas y Regreso de llamadas como paquetes. Nuevamente, dentro de cada paquete, tendrá un conjunto de casos de uso.

# Taller

Verifique su conocimiento en la recopilación de necesidades y localice las respuestas en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿Cómo hemos representado las necesidades del sistema?
2. ¿Una vez que se hace el análisis del dominio ya finaliza el modelado de clases?
3. ¿Qué es el "tiempo muerto"?

## Ejercicio

Continúe con el dominio de la Biblioteca. ¿Cuáles son los principales paquetes de funcionalidad? ¿Cuáles son los casos de uso que los componen?

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 19

# Desarrollo de los casos de uso

Ahora que ha visto lo que está involucrado en la recopilación de necesidades, veremos el análisis de tales necesidades y las llevaremos a cabo en un sistema.

En esta hora se tratarán los siguientes temas:

- Cuidado y provisión de casos de uso
- La especificación de descripciones, condiciones previas y resultantes.
- La especificación de pasos
- La diagramación de los casos de uso

Los casos de uso del diagrama de paquetes de la hora 18, "Recopilación de las necesidades del sistema", nos dio una buena idea de lo que el sistema tendrá que hacer. El equipo tendrá que llevar a cabo cada uno de ellos. Poco a poco han pasado de comprender el dominio a comprender el sistema. Los casos de uso establecieron el puente.

Si cree que el proyecto del desarrollo del sistema se orienta a los casos de uso, ya habrá comprendido todo el proceso.

www.elsolucionario.net

www.elsolucionario.net

Observe que en ningún momento de la sesión JAD el equipo de desarrollo habló de la forma en que el sistema realizaría todas las actividades indicadas en los diversos casos de uso. La idea fue la de enumerar todos los casos de uso posibles. Conforme se realicen los casos de uso en esta hora, verá la forma en que los componentes del sistema RICO empezarán a materializarse. En este punto del desarrollo, el sistema empieza a ser el centro de atención.

Vamos a ponernos en los zapatos del equipo de desarrollo y veremos algunos de los casos de uso obtenidos.

# Cuidado y provisión de los casos de uso

Para analizar a los casos de uso, tendremos que realizar otra sesión JAD. El debate en esta sesión se orienta a derivar un análisis para cada caso de uso.

A modo de advertencia: la sesión JAD de los casos de uso es, por lo general, la más compleja, dado que pide a los participantes (usuarios potenciales del sistema terminado) que se conviertan en analistas. En su propio nicho, cada uno es un experto del dominio, y tendrá que profundizar en su experiencia. Por lo general, no están acostumbrados a expresar o analizar su conocimiento. De hecho, también es probable que tampoco hayan formado parte de algún proyecto de diseño, y que no se sientan muy a gusto al intentar especificar lo que un sistema debería hacer para ayudarles en su trabajo.

Para reducir estas presiones, es mejor organizar la sesión JAD para que el equipo trate con un grupo a la vez; por ejemplo: sólo los meseros. De esa forma, los demás no estarán ociosos mientras los meseros analizan sus casos de uso. Quienes conocen las generalidades del dominio, los restauranteros, podrían dar una mano en todos los grupos. Una selección de usuarios sería adecuada cuando se trate el tema del paquete del Cliente.

Son muchos los casos de uso, y para no extendernos mucho en esta hora, nos enfocaremos en los primeros nueve de ellos, propios del paquete Mesero. Luego de que vea cómo se harán tales análisis, podrá hacer el resto de los casos de uso de los Meseros, así como los de los demás paquetes.

# El análisis de los casos de uso

Por lo que se mencionó anteriormente (hora 7, "Diagramas de casos de uso"), recuerde que, cada caso de uso es una colección de situaciones, y cada una de éstas es una secuencia de pasos. Para cada escenario en cada caso de uso, queremos mostrar:

- Una breve descripción del escenario
- Conjeturas del escenario

www.elsolucionario.net

www.elsolucionario.net

- El actor que inicia el caso de uso
- Condiciones previas para el caso de uso
- Pasos relacionados con el sistema en el escenario
- Condiciones resultantes una vez terminado el escenario
- El actor beneficiado del caso de uso

En su análisis también podrá incluir cualquier condición excepcional o alternativas. No obstante, los presentes escenarios se han mantenido simples.

Ninguna forma específica es "correcta" para establecer un análisis de casos de uso. Los elementos que he listado establecen, por lo general, un panorama del caso de uso.

En su documento de diseño (el que le da a su cliente y programadores), cada uno de estos análisis de casos de uso deberán estar en páginas por separado. Posiblemente quiera incluir un diagrama del caso de uso, con todo y actores, en esta página.

Los pasos, en el escenario, que están relacionados con el sistema son muy importantes. Mostrarán la forma en que se supone que funcionará el sistema. Cuando los participantes de la sesión JAD nos digan los pasos, de hecho nos estarán diciendo cómo va a lucir el sistema. Después de esta sesión JAD, deberíamos tener una firme idea de los componentes del sistema.

También es importante hacer conjeturas. En la lista de conjeturas, podrá indicar algunas consideraciones de diseño, como lo verá posteriormente.

Esto era lo que quise decir cuando dije que el proyecto sería "orientado a casos de uso". Los casos de uso crearán, a fin de cuentas, la ruta de acceso al sistema.

# El paquete Mesero

La clase Mesero se distingue por ser la de mayor actividad, lo cual no debería sorprendernos, dado que el mesero interactúa prácticamente con las demás clases.

www.elsolucionario.net

www.elsolucionario.net

Los casos de uso del Mesero son:

- Tomar una orden
- Transmitir la orden a la cocina
- Cambiar una orden
- Recibir una notificación de la cocina
- Sondear el progreso de la orden
- Notificar al chef del progreso de los clientes en sus alimentos
- Totalizar una cuenta
- Imprimir una cuenta
- Llamar a un asistente
- Llamar a un mozo de piso
- Tomar una orden de bebida
- Transmitir una orden de bebida al bar
- Obtener un acuse de recibo
- Recibir una notificación del área del bar

## Tomar una orden

Empecemos con "Tomar una orden". Vamos a preguntarle a los meseros experimentados, para que nos den una descripción, conjeturas, condiciones previas, pasos y condiciones resultantes. El paquete y subpaquete ya indica al actor que inicia (Mesero) y al beneficiado (Cliente).

Una buena descripción en una sola frase podría ser: "El mesero captura la orden del cliente en la palmtop y la transmite a la cocina." Se asume que un cliente desea un platillo, que ha leído el menú y que ya hizo su elección. También se puede asumir que la palmtop del mesero cuenta con una interfaz para capturar órdenes.

Las condiciones previas son que un cliente se haya sentado y haya leído el menú. La condición resultante es que tal pedido se haya capturado en RICO.

Los pasos en el caso de uso son:

1. En la palmtop, el Mesero activa la interfaz para capturar una orden.
2. Aparece la interfaz para capturar órdenes.
3. El Mesero registra la selección del Cliente hecha del menú en RICO.
4. El sistema transmite la orden a la PC de la cocina.

www.elsolucionario.net

www.elsolucionario.net

Aunque hemos asumido que ya hay una interfaz para registrar órdenes, aún no hemos especificado cómo luciría o cómo sería el proceso físico de registrar la orden. Todavía no sabemos cómo lucirá la interfaz de la PC de la cocina, ni hemos dicho nada de los detalles técnicos de transmitir una orden.

El punto es que, conforme ponemos de manifiesto nuestras conjeturas en cuanto al diseño, empezamos a tener una idea de lo que se supone que debe hacer el sistema y empezamos a dar forma a nuestras ideas para ponerlo en práctica. Los pasos en los casos de uso nos fuerzan a asumir la idea acerca de los componentes del sistema. Recuerde que los casos de uso pretenden mostrar como se ve el sistema para un usuario.

## Transmitir la orden a la cocina

¿Listo para otro? Éste será incluido en (esto es, "usado por") al menos dos casos de uso: la anterior y "Cambiar una orden".

La descripción es: "Tomar una orden que ha sido registrada en la palmtop, colocarla en la red inalámbrica y enviarla a la PC de la cocina". Se asume que ya tenemos medios para comunicar la orden (mediante una red inalámbrica) y, nuevamente, que contamos con una interfaz para registrar una orden. ¿Tenemos que repetir esta conjetura? Sí. Cada caso de uso aparecerá eventualmente en una página por separado en el documento de diseño, lo que servirá como una referencia del sistema. Para mayor claridad, las conjeturas deberían aparecer en cada caso de uso, aun si tenemos que repetirlas en cada uno de ellos.

La condición previa es una orden registrada en una palmtop. La condición resultante es que la orden llegue a la cocina. El actor beneficiado es el Cliente.

Los pasos son:

1. Hacer clic en un botón en la interfaz para el registro de órdenes indica "Enviar a la cocina".
2. RICO transmitirá la orden mediante la red inalámbrica.
3. La orden llega a la cocina.
4. La interfaz del usuario para registrar las órdenes en la palmtop indica que la orden ha llegado a la cocina.

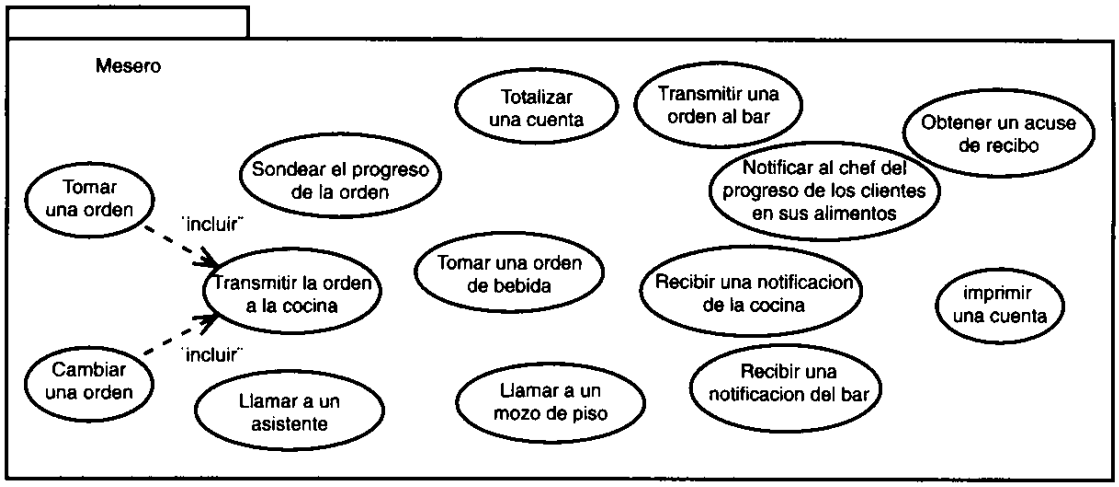
Claro está que tendremos que modificar nuestro diagrama de casos de uso para el subpaquete Cliente. Tiene que mostrar la dependencia «incluir» entre este caso de uso y "Tomar una orden", y entre este caso de uso y "Cambiar una orden". La figura 19.1 muestra los diagramas actualizados del caso de uso para el paquete Mesero.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 19.1**

*Los diagramas actualizados de casos de uso para el paquete Mesero.*




## Cambiar una orden

Dentro de este contexto, vayamos a "Cambiar una orden". La descripción es "Modificar una orden ya registrada en RICO". Se asume que ya se ha registrado y enviado a la cocina una orden y que, subsecuentemente, el cliente desea cambiarla. También asumiremos que RICO cuenta con una base de datos de órdenes que muestra al mesero quién ha capturado cada orden y la mesa de donde provino, que el mesero podrá acceder a tal base de datos en la palmtop, que RICO puede transmitir y recibir de la palmtop a la PC de la cocina, y que la palmtop tiene una interfaz de usuario capaz de cambiar una orden.

La condición previa es la orden registrada con anterioridad. La condición resultante es que la orden modificada haya llegado a la cocina. El actor beneficiado es el Cliente.

Los pasos en este caso de uso son:

1. En la palmtop, el mesero activa la interfaz del usuario para cambiar una orden.
2. La interfaz del usuario trae una lista de órdenes realizadas y enviadas a la cocina por el mesero.
3. El mesero selecciona la orden por cambiar.
4. El mesero registra la modificación a la orden.
5. El sistema transmite la orden a la PC de la cocina.

El paso 5 incluye al caso de uso anterior "Transmitir la orden a la cocina".

www.elsolucionario.net

www.elsolucionario.net

## Sondeo del progreso de la orden

Como podrá recordar, los debates anteriores respecto al restaurante del futuro incluyeron la posibilidad de descubrir en qué momento la orden de un cliente saldría de la cocina. Este caso de uso hace exactamente eso. Si se instaura en el sistema se facilitará enormemente el trabajo del mesero.

La descripción es: "Sondear el progreso (el tiempo para completar) de una orden que ya se haya registrado en RICO". Se asume que ya se ha realizado, registrado y enviado una orden a la cocina, y que el cliente desea saber cuánto más esperará para que llegue su platillo. Repetimos dos conjeturas de diseño anteriores: una base de datos de pedidos y la facultad de transmitir mensajes entre la palmtop y la PC de la cocina. También asumimos una interfaz del usuario en la palmtop para el sondeo de las órdenes y otra en la PC de la cocina con el mismo propósito.

La condición previa es la orden registrada con anterioridad. La condición resultante es que el estado de la orden ha llegado a la palmtop del mesero. El actor beneficiado es el Cliente.

Los pasos son:

1. El mesero activa la interfaz en la palmtop para sondear una orden registrada.
2. La interfaz le muestra al mesero una lista de las órdenes que tiene registradas en la cocina.
3. El mesero elige la orden que desea sondear.
4. El sistema transmite el mensaje de sondeo a la PC de la cocina.
5. La PC de la cocina recibe el mensaje.
6. El chef selecciona la orden de la cual se quiere conocer su avance.
7. El chef teclea un tiempo estimado para completar la orden.
8. El sistema transmite el tiempo estimado a la palmtop del mesero.

www.elsolucionario.net

## Notificar al chef del progreso de los clientes en sus alimentos

A partir de este caso de uso, utilizaré subtítulos para indicar los aspectos del análisis del caso de uso y viñetas para establecer frases para cada subtítulo con dos excepciones: Seguiré numerando los pasos y no utilizaré viñetas para la descripción.

### Descripción

Mediante la red, el mesero le indica al chef que un cliente ya casi ha finalizado con su entremés.

www.elsolucionario.net

## Conjeturas

- El mesero se encuentra en el área de servicio del cliente
- El mesero puede medir el progreso del cliente
- El sistema cuenta con una interfaz para el estado del cliente
- El sistema transmite mensajes de la palmtop a la PC de la cocina y viceversa.

## Condiciones previas

- El cliente ha finalizado de manera parcial el entremés

## Condiciones resultantes

- El chef ha iniciado los pasos finales para completar el plato principal

## Pasos

1. En la palmtop, el mesero activa la interfaz para el estado del cliente.
2. La interfaz de usuario trae una lista de las mesas en el área de servicio del mesero.
3. El mesero elige la mesa indicada.
4. El mesero envía un mensaje de "ya casi terminó con el entremés" a la PC de la cocina.
5. La PC de la cocina recibe el mensaje.
6. El mesero obtiene un acuse de recibo de la PC de la cocina.

Este último paso utiliza el caso de uso "Acuse de recibo", que se encuentra en el paquete Mesero. La figura 19.2 le muestra un diagrama de este caso de uso.

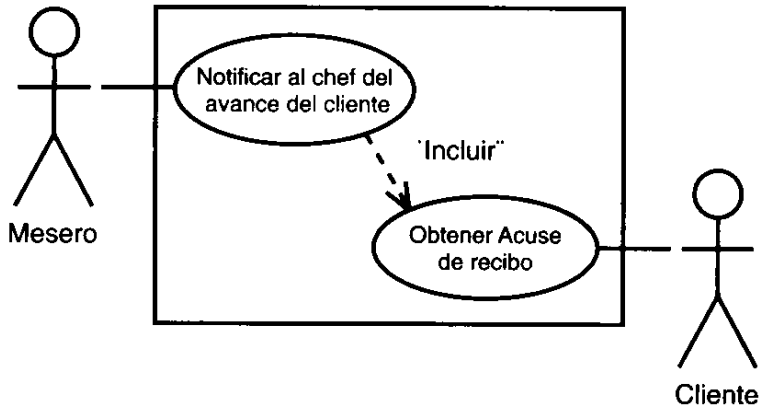



**FIGURA 19.2**

*Diagrama de caso de uso para "Notificar al chef del progreso de los clientes en sus alimentos".*

www.elsolucionario.net

www.elsolucionario.net

### Actor beneficiado

- Cliente

## Totalizar una cuenta

### Descripción

Sumar los elementos en la orden.

### Conjeturas

- Existe una base de datos de pedidos accesibles mediante la palmtop
- Cada elemento de la orden se relaciona con su precio

### Condiciones previas

- El servicio ya ha terminado con sus alimentos

### Condiciones resultantes

- Se totaliza la cuenta

### Pasos

1. El mesero trae una lista de órdenes activas en la palmtop.
2. El mesero elige la orden adecuada.
3. El mesero presiona un botón en la palmtop para totalizar la cuenta.
4. El sistema calcula el total de los precios en la cuenta.

### Actor beneficiado

- Cliente

## Imprimir una Cuenta

Aunque podría parecer trivial, es una parte importante de la transacción.

### Descripción

Imprimir la cuenta totalizada.

### Conjeturas

- Una impresora conectada a la red inalámbrica en el área de servicio

### Condiciones previas

- Una cuenta totalizada

www.elsolucionario.net

www.elsolucionario.net

### Condiciones resultantes

- Una cuenta impresa

## Pasos

1. El mesero presiona un botón para imprimir la cuenta.
2. La impresora conectada a la red en el área de servicio imprime la cuenta.
3. El mesero presiona un botón en la palmtop para quitar esta orden de la lista de órdenes activas.

### Actor beneficiado

- Cliente

# Llamar a un Asistente

## Descripción

Solicitar a un asistente que limpie la mesa para el siguiente cliente.

## Conjeturas

- El sistema permite una comunicación inalámbrica entre dos empleados en movimiento
- El sistema cuenta con una interfaz para enviar un mensaje a un asistente

## Condiciones previas

- Una mesa vacía tendrá que ser limpiada y preparada

## Condiciones resultantes

- El asistente llega a la mesa para limpiarla y prepararla

## Pasos

1. El mesero activa la interfaz para enviar un mensaje a un asistente.
2. El mesero obtiene un acuse de recibo del asistente.

Como en el caso de uso "Notificar al chef del progreso de los clientes en sus alimentos", el último paso utiliza el caso de uso "Acuse de recibo".

## Actor beneficiado

- Asistente

www.elsolucionario.net

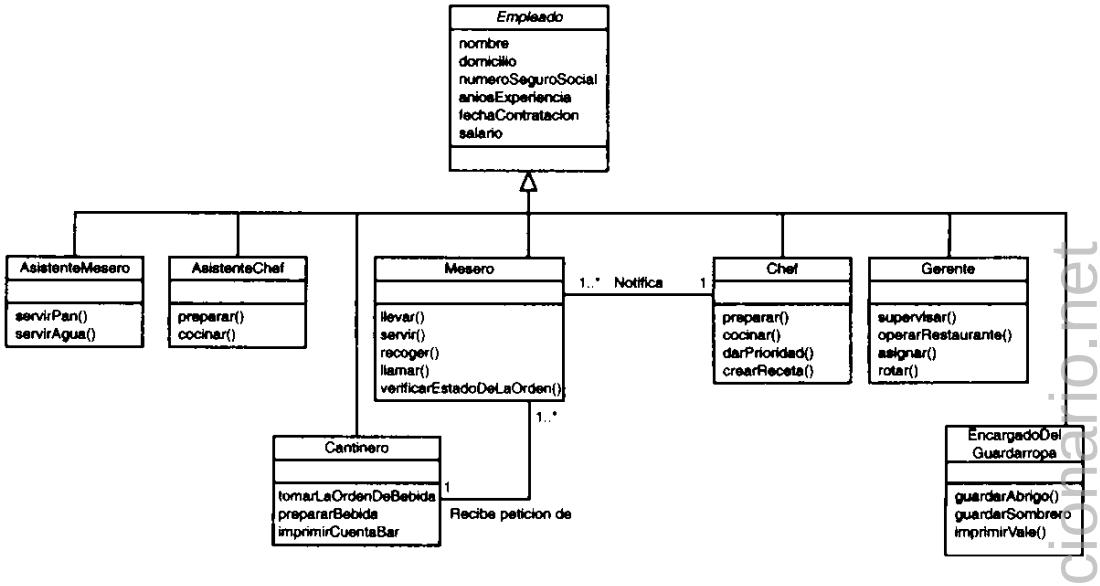
www.elsolucionario.net

Al analizar este caso de uso, así como los casos de uso en el paquete del Asistente, tal vez pensemos que separar la clase Asistente en dos, AsistenteMesero y AsistenteChef, sería una buena idea (aclararía las cosas). ¿Podrían ambas ser clases secundarias de una clase abstracta Asistente? Podrían, pero posiblemente no se obtendría mucho de establecer esta clase abstracta.

La creación de estas dos clases requiere una revisión al análisis del dominio. Tendremos que volver a trabajar los diagramas de clases, en particular el diagrama de Empleado, como muestra la figura 19.3.

**FIGURA 19.3**

*El diagrama actualizado de la clase Empleado.*




También tendríamos que actualizar nuestros diagramas de paquetes para incluir el de Asistente Mesero y Asistente Chef.

Esto es un ejemplo de la forma en que los segmentos de GRAPPLE se intercomunican. Los conocimientos obtenidos durante el análisis de los casos de uso han ayudado a desarrollar el análisis del dominio.

## Casos de uso restantes

Los restantes casos de uso en el paquete Mesero son muy similares a los que hemos analizado. Le dejaré como ejercicio finalizar el análisis de este paquete (vea el ejercicio 1 del taller).

# Componentes del sistema

Un aspecto importante del análisis del caso de uso es que empezó a descubrir componentes del sistema. Antes de finalizar con esta hora, tome nota de los componentes que hemos visto durante nuestro análisis de los casos de uso en el paquete Mesero. Los verá en la sección Conjeturas de cada análisis de los casos de uso (habrá otros componentes que aparecerán cuando haga los ejercicios).

www.elsolucionario.net

www.elsolucionario.net

En el lado del software, es obvio que son necesarias varias interfaces de usuario. RICO necesitará interfaces del usuario basadas en palmtops para registrar órdenes, cambiarlas, sondearlas, dar el estado del cliente y enviar mensajes para un asistente. Como buena medida, algo como una interfaz tipo "página principal" será necesario para tener organizadas las demás interfaces. RICO también necesitará una interfaz del usuario en la PC de la cocina para permitir al chef verificar el avance de una orden.

También parece que necesitaremos una base de datos que contenga a todos los pedidos. Cada registro contendrá la mesa, el pedido, la hora en que se realizó, el mesero, si el pedido está activo, y cosas así.

Del lado del hardware, necesitaremos una red inalámbrica, palmtops para los empleados en movimiento (meseros, asistentes y mozos de piso) y una PC de escritorio en la cocina, así como otra en la recepción. Necesitaremos una impresora conectada a la red en cada área de servicio. Probablemente, también necesitaremos una palmtop y una impresora para el encargado del guardarropa.

Ya está tomando forma un mejor documento de diseño. En la hora siguiente, profundizará aún más en los casos de uso.

# Resumen

No basta con listar todos los casos de uso. Un equipo de desarrollo deberá comprender cada uno detalladamente para empezar a comprender el sistema. En esta hora vimos detalladamente el análisis de cada parte de un caso de uso.

Un análisis de casos de uso involucra la especificación de una descripción del propio caso de uso, derivar las condiciones previa y resultante, y especificar los pasos. Un aspecto importante del análisis de los casos de uso es que los componentes del sistema empiezan a ser evidentes.

# Preguntas y respuestas

**P En el segmento inicial de GRAPPLE, noté que pasó por alto la acción "Identificación de los sistemas cooperativos". ¿A qué se debe?**

**R** Como recuerda, este equipo de desarrollo inició un proyecto sin precedentes. No hay sistemas cooperativos. No obstante, el siguiente sistema que alguien diseñe para Restaurantes LNG tal vez tendría que acceder a RICO de alguna forma.

**P En esta hora modificó los diagramas de casos de uso y el de clases. ¿Esto sucede con frecuencia?**

**R** Así es. No deberá ser reacio a hacer modificaciones conforme evolucione su conocimiento. La lista original de casos de uso capturó todo el conocimiento en cierto punto del proyecto, y captura una imagen de tal momento. Los diagramas modificados representarán las ideas subsecuentes del equipo de desarrollo.

www.elsolucionario.net

www.elsolucionario.net

# Taller

El taller para esta hora verificará su conocimiento para realizar casos de uso. Para ver las respuestas reales, vea el apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿Cuáles son las partes de un diagrama de casos de uso típico?
2. ¿A qué se refiere que un caso de uso "incluya" (o "utilice") a otro?

## Ejercicios

1. Genere el diagrama de casos de uso para "Llamar a un mozo de piso".
2. Analice los restantes casos de uso del paquete Mesero, y dibuje sus diagramas.
3. Analice los casos de uso del paquete Chef, y dibuje los casos de uso.
4. Haga lo mismo para los paquetes Cantinero, Asistente y Mozo de piso.

Analice los casos de uso que haya obtenido de su proyecto Biblioteca.

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 20

# Orientación a las interacciones y cambios de estado

A continuación profundizará en los casos de uso analizados en la hora anterior. Los utilizará como fundamento para comprender las partes del sistema y la forma en que se comunican entre sí.

En esta hora se tratarán los siguientes temas:

- Las partes funcionales del sistema
- Modificación de los casos de uso
- Análisis de la colaboración entre las partes funcionales

El análisis de los casos de uso de la hora anterior es un gran avance para hacer realidad el sistema RICO. No obstante, el análisis aún está muy lejos de permitir que se empiece con la codificación.

Analizar los casos de uso ha ayudado a visualizar las partes funcionales del sistema. Aunque ahora sabemos mucho de ellos, aún tenemos que modelar la forma en que las partes funcionales se comunicarán entre sí y la forma (y momento) en que cambiarán de estado. Dar esta información a los programadores facilitará su trabajo. Tendrán una visión de la forma de codificar las clases y hacer que trabajen en conjunto.

www.elsolucionario.net

www.elsolucionario.net

# Las partes funcionales del sistema

Una forma de empezar es enumerar los componentes sugeridos del sistema en cada paquete de casos de uso. Aunque no hemos analizado de manera explícita todos los casos de uso en todos los paquetes en la hora anterior, aún podríamos obtener los componentes que se asumen en ellos. Claro que en un verdadero proyecto de desarrollo, un equipo de desarrollo debería analizar todos los casos de uso antes de continuar.

## El paquete Mesero

Al finalizar la hora anterior, enumeramos las partes de software del sistema de acuerdo con nuestro análisis de los primeros nueve casos de uso del paquete Mesero: en las palmtops, RICO necesitará interfaces para el registro de órdenes, su modificación, su control, el estado de los clientes y el envío de mensajes. También será necesaria una pantalla principal en la interfaz. Nuestro análisis trajo a la luz la necesidad de una interfaz para ver el avance de los pedidos en la PC de la cocina. RICO necesitará una base de datos para contener a todas las órdenes.

Los casos de uso que no analizamos también sugieren componentes del sistema. Para refrescar su memoria, esos casos eran:

- Llamar a un mozo de piso
- Tomar una orden de bebida
- Transmitir una orden de bebida al bar
- Obtener un acuse de recibo
- Recibir una notificación del bar

Los casos de uso sugieren algunos componentes evidentes. El primero nos indica que algo en la interfaz del Mesero (como una pantalla dedicada) tiene que permitir la llamada a un mozo de piso. El segundo, que se necesita una pantalla para obtener una orden de bebidas (similar a la que se usa para tomar una orden de alimentos). La interfaz debe poder obtener un acuse de recibo (para mostrar, por decir, que el mozo de piso ha recibido la petición) y recibir un mensaje del bar que indique que está lista la bebida.

Debido al trabajo de un mesero, no debe sorprendernos que los componentes primordiales en este paquete son interfaces relacionadas con el registro de órdenes, así como la recepción y envío de mensajes.

www.elsolucionario.net

www.elsolucionario.net

## El paquete Chef

Los casos de uso del paquete Chef son:

- Almacenar una receta
- Obtener una receta
- Notificar al mesero
- Recibir una petición del mesero
- Dar acuse de recibo a una petición del mesero
- Indicar el periodo de preparación
- Asignar una orden

¿Qué componentes sugieren estos casos de uso? Nuevamente, algunos vienen a la mente de manera evidente.

## El paquete Mozo De Piso

Los casos de uso para el mozo de piso son:

- Recibir una petición del mesero
- Dar acuse de recibo a una petición
- Indicar que una mesa ha sido limpiada

## El paquete Asistente Mesero

Como recordará, decidimos dividir el paquete Asistente en Asistente Mesero y Asistente Chef. Los casos de uso del Asistente Mesero serían:

- Obtener una petición del mesero
- Dar acuse de recibo a la petición
- Notificar que se ha completado la petición

## El paquete Asistente Chef

Los casos de uso del Asistente Chef serían:

- Obtener una petición del chef
- Dar acuse de recibo a la petición
- Notificar que se ha completado la petición

www.elsolucionario.net

www.elsolucionario.net

Tal vez podría pensarse que no es necesaria una computadora para un asistente de chef dado que trabaja de forma muy cercana al chef en la cocina. No obstante, si la cocina es muy grande, la comunicación electrónica podría ser una buena idea.

## El paquete Cantinero

Los casos de uso del Cantinero son:

- Capturar la receta de una bebida
- Obtener la receta de una bebida
- Recibir una notificación del mesero
- Recibir una petición del mesero
- Dar acuse de recibo de una petición
- Notificar que la petición se ha completado

Estos casos de uso son similares a los del paquete del Chef, y los componentes de software que sugieren también lo son. El hardware es similar, a su vez: detrás de una barra, una PC de escritorio podría tener más sentido que una palmtop.

Necesitamos una base de datos de recetas para bebidas y una interfaz que permitan un fácil acceso a ella (tanto para capturar como para obtener recetas). La interfaz para el cantinero tiene que mostrar una notificación proveniente del mesero (de que la mesa de un cliente está lista) y una petición de una bebida también proveniente del mesero. El cantinero tendrá que enviar un acuse de recibo de que la petición se ha recibido y, también, notificar al mesero que la bebida está lista.

## El paquete Encargado Del Guardarropa

Los casos de uso del Encargado Del Guardarropa son:

- Imprimir un vale de abrigo
- Imprimir un vale de sombrero

Los componentes de software en la palmtop del encargado del guardarropa deberían incluir una interfaz que permita la impresión del vale adecuado. El vale deberá incluir la hora y una descripción del artículo. Posiblemente también queramos tener una base de datos de elementos almacenados.

# Colaboración en el sistema

En este punto del proyecto, la tarea será mostrar la forma en que los componentes interactúan para completar cada caso de uso. Modelaremos tales interacciones para un par de casos de uso en el paquete del Mesero. El conjunto de casos de uso es demasiado grande para que los veamos todos; pero en un proyecto real, un equipo de desarrollo hará justamente eso.

www.elsolucionario.net

www.elsolucionario.net

Recuerde que, como dije antes: Detrás de cada caso de uso se esconde un diagrama de interacciones.

## Tomar una orden

Empecemos con el caso de uso "Tomar una orden". De lo desprendido en la hora 19, los pasos son:

1. En la palmtop, el Mesero activa la interfaz para capturar una orden.
2. Aparece la interfaz para capturar órdenes.
3. El Mesero registra la selección del Cliente hecha del menú en RICO.
4. El sistema transmite la orden a la PC de la cocina.

En el modelo que desarrollamos en la hora anterior, este caso de uso incluye a "Transmitir la orden a la cocina", cuyos pasos son:

1. Hacer clic en un botón en la interfaz para el registro de órdenes indica "Enviar a la cocina".
2. RICO transmitirá la orden mediante la red inalámbrica.
3. La orden llega a la cocina.
4. La interfaz del usuario para registrar las órdenes en la palmtop indica que la orden ha llegado a la cocina.

Un diagrama de secuencias podría mostrar bastante bien esta colaboración (igual lo haría un diagrama de colaboraciones, mismo que le pediré que genere en el ejercicio 1). La preparación del diagrama nos fuerza a enfocar nuestra imaginación de varias formas.

Para empezar, cuando el mesero toma la orden de un cliente, creará algo: ¡una orden! Tal orden es un objeto en el sistema RICO (a su vez, es una instancia de una clase, Orden, proveniente de nuestro análisis del dominio en la hora 17, "Elaboración de un análisis del dominio"). El chef la usará como orientación para iniciar y realizar un conjunto de acciones. El mesero totalizará una cuenta de acuerdo con ella. El cliente pagará la cuenta. Así, esta orden generada es un elemento importante.

A su vez, si examina los casos de uso "Cambiar una orden" y "Sondear el progreso de la orden" (como lo haremos en un momento), verá que se hace referencia a una lista de órdenes. Esta lista se habrá obtenido de una base de datos de órdenes, misma que mencioné al finalizar la hora 19. ¿Cómo se captura la orden en la base de datos? Bien, pues ello deberá ocurrir en este caso de uso.

www.elsolucionario.net

www.elsolucionario.net

Podemos enfocar nuestra imaginación de otra forma. En el caso de uso incluido, el término "cocina" es un poco vago. Dado que estamos modelando componentes de software, tendremos que pulir lo que queremos decir aquí. Imaginar cómo podría funcionar todo nos llevará a una forma donde el sentido común establecerá que la orden deberá mostrarse de algún modo en la interfaz del chef en la PC de la cocina. El cómo ocurra no es algo que en este momento deberá ocuparnos.

Luego de que hemos pensado en lo anterior, el caso de uso "Tomar una orden" luciría así:

1. En la palmtop, el Mesero activa la interfaz para capturar una orden.
2. Aparece la interfaz para capturar órdenes.
3. El Mesero registra en RICO la selección del Cliente hecha del menú.
4. RICO genera una orden.
5. Incluir(Transmitir una orden a la cocina).
6. El sistema guarda la orden en la base de datos de órdenes.

"RICO" y "el sistema" son sinónimos.

Observe el recurso utilizado en el paso 5 que indica la inclusión del caso de uso "Transmitir una orden a la cocina".

La "base de datos de pedidos" emplaza nuestra imaginación hacia el lado del hardware. Esta base de datos debe encontrarse en una computadora, pero no hemos establecido una. Una posibilidad sería contar con un equipo de cómputo central que tuviera esta base de datos y la hiciera accesible a las demás máquinas de la red.

Mientras estamos en esto, debemos expandir el caso de uso "Transmitir la orden a la cocina" e incluir la modificación respecto a la interfaz del chef. Dado que tiene un paso que establece un mensaje en la palmtop del mesero que indica la recepción de la orden en la cocina, debemos agregar un paso que haga que el sistema envíe un mensaje de la PC de la cocina a la palmtop. Así, los pasos serían:

1. Hacer clic en un botón en la interfaz para el registro de órdenes indica "Enviar a la cocina".
2. RICO transmitirá la orden mediante la red inalámbrica.
3. La orden llegará a la interfaz del chef en la PC de la cocina.
4. RICO enviará un mensaje de la cocina a la interfaz de la palmtop que acuse el recibo de la orden.
5. La interfaz del usuario, para registrar las órdenes en la palmtop, indicará que la orden ha llegado a la cocina.

www.elsolucionario.net

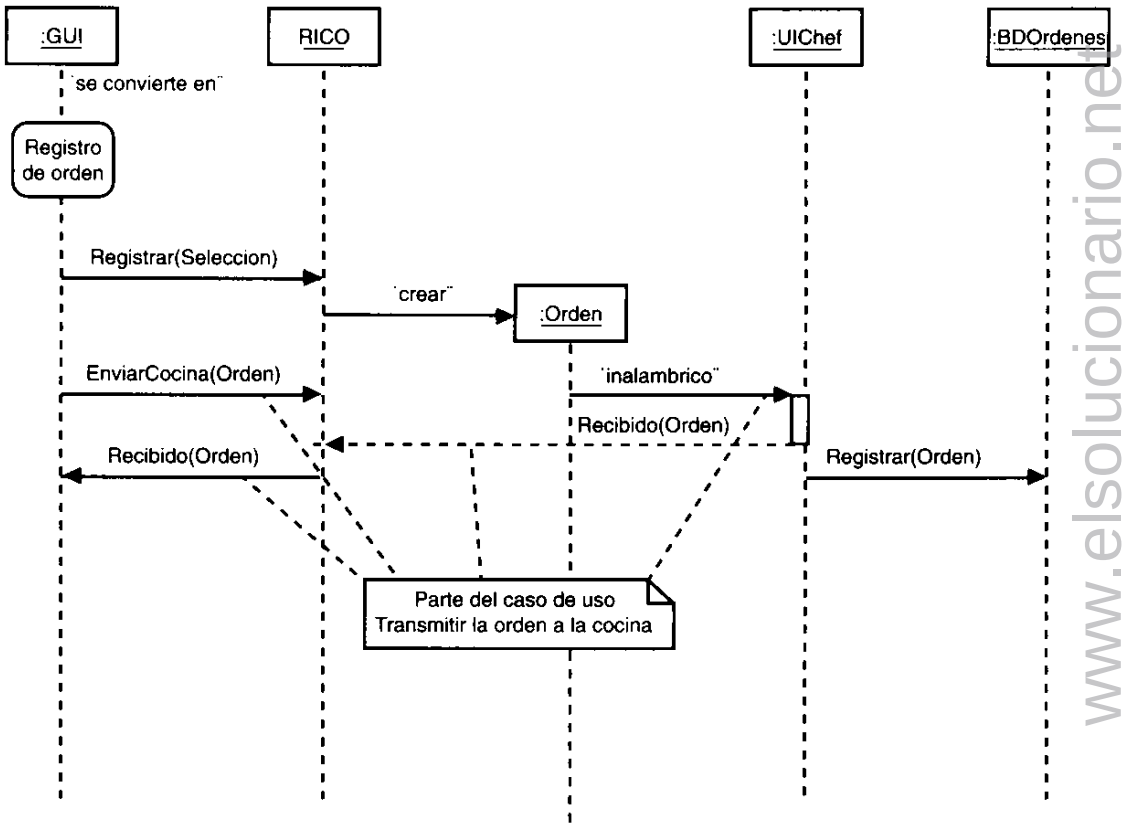
www.elsolucionario.net

Estas modificaciones a los casos de uso son sólo dos ejemplos más de la forma en que una fase de un proyecto puede influenciar a otra. En este contexto, la preparación de nuestros diagramas de secuencias nos ayudaron a pulir nuestra idea de los casos de uso con la base de esos diagramas.

La figura 20.1 le muestra el diagrama de secuencias que captura nuestra idea de este caso de uso. Para recordar lo que vio de los diagramas de secuencias, los objetos que están colocados en la parte superior de él representan a los componentes en este caso de uso. El objeto Orden se genera durante el caso de uso, y por ello está bajo los otros dos. El mensaje que apunta hacia él tiene un estereotipo «crear». La línea punteada que parte de cada objeto representa la "línea de vida de un objeto", misma que avanza hacia abajo verticalmente. Los pequeños rectángulos en los tiempos de actividad se llaman "activaciones". Cada activación representa al periodo en el que un objeto realiza una acción.

**FIGURA 20.1**

*El diagrama de secuencias de "Tomar una orden".*




Vea el cambio de estado en el primer tiempo de actividad. Intenta aclarar la forma en que una interfaz maneja un tipo especial de actividad. Podríamos incluir todos los cambios de estado posibles como diagramas de estados por separado, pero sería excesivo. El colocarlos en diagramas de secuencias (al menos en este dominio) aparentemente es más sencillo.

En un diagrama de secuencias, una flecha de mensaje con una línea punteada representa a un mensaje devuelto.

www.elsolucionario.net

www.elsolucionario.net

## Cambiar una orden

Intentemos con otra. En la hora anterior, los pasos de caso de uso "Cambiar una orden" fueron:

1. En la palmtop, el mesero activa la interfaz del usuario para cambiar una orden.
2. La interfaz del usuario trae una lista de órdenes realizadas y enviadas a la cocina por el mesero.
3. El mesero selecciona la orden por cambiar.
4. El mesero registra la modificación a la orden.
5. El sistema transmite la orden a la PC de la cocina.

Nuevamente, la preparación del diagrama nos ayuda a pulir nuestra imaginación y modificar de cierto modo al caso de uso. Luego del paso 4, sin duda queremos que el sistema cree una orden modificada. Luego del paso 5, el sistema debería registrar la orden modificada en la base de datos de órdenes.

Con ello, el nuevo caso de uso debería ser:

1. En la palmtop, el mesero activa la interfaz del usuario para cambiar una orden.
2. La interfaz del usuario trae una lista de órdenes realizadas y enviadas a la cocina por el mesero.
3. El mesero selecciona la orden por cambiar.
4. El mesero registra la modificación a la orden.
5. RICO crea una orden de acuerdo con la modificación a la anterior.
6. Incluir(Transmitir una orden a la PC de la cocina).
7. El sistema guarda la orden modificada en la base de datos de órdenes.

Nuevamente, utilizamos el recurso para indicar la inclusión de un caso de uso.

La figura 20.2 le muestra el diagrama de secuencias que corresponde a este caso de uso.

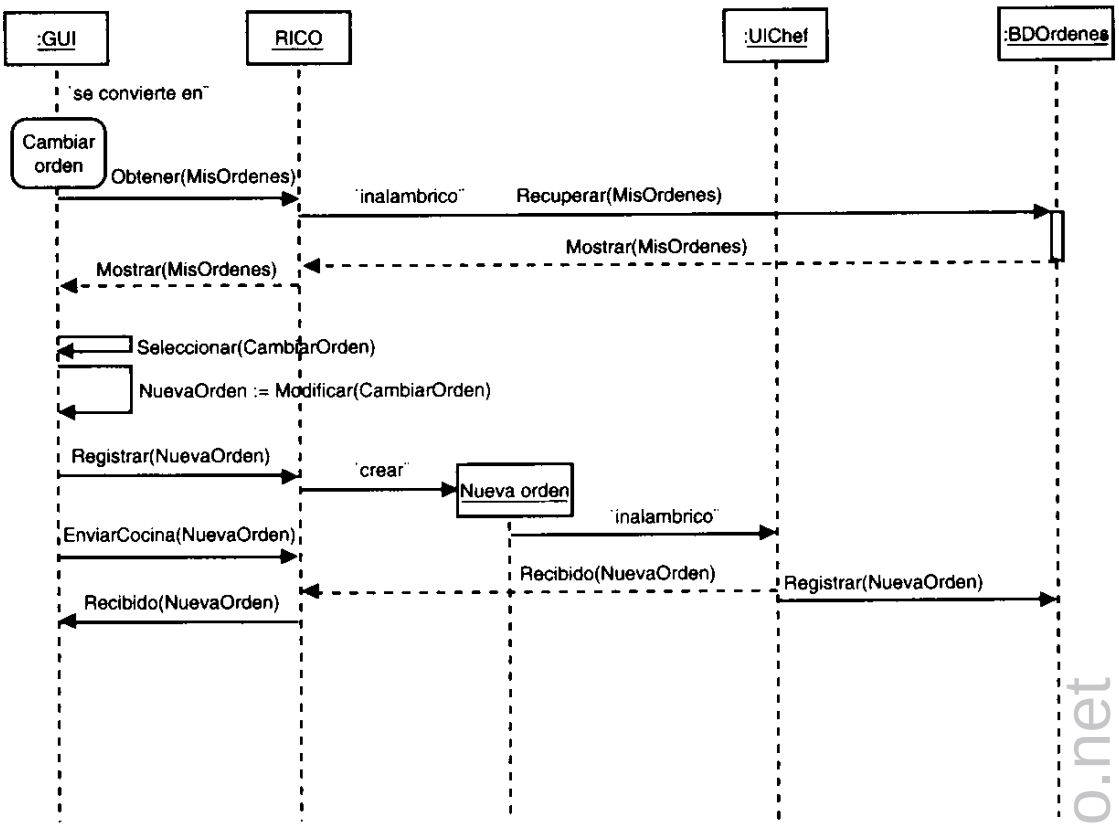
Al igual que en la figura 20.1, ahora mostramos un cambio de estado.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 20.2**

*El diagrama de secuencias para "Cambiar una orden".*




## Sondeo del progreso de la orden

Veamos otro caso de uso antes de terminar. El caso de uso "Sondear el progreso de la orden" consta de los siguientes pasos:

1. El mesero activa la interfaz en la palmtop para sondear una orden registrada.
2. La interfaz le muestra al mesero una lista de las órdenes que tiene registradas en la cocina.
3. El mesero elige la orden que desea sondear.
4. El sistema transmite un mensaje de sondeo a la PC de la cocina.
5. La PC de la cocina recibe el mensaje.
6. El chef selecciona la orden de la cual se quiere conocer su avance.
7. El chef teclea un tiempo estimado para completar la orden.
8. El sistema transmite el tiempo estimado a la palmtop del mesero.

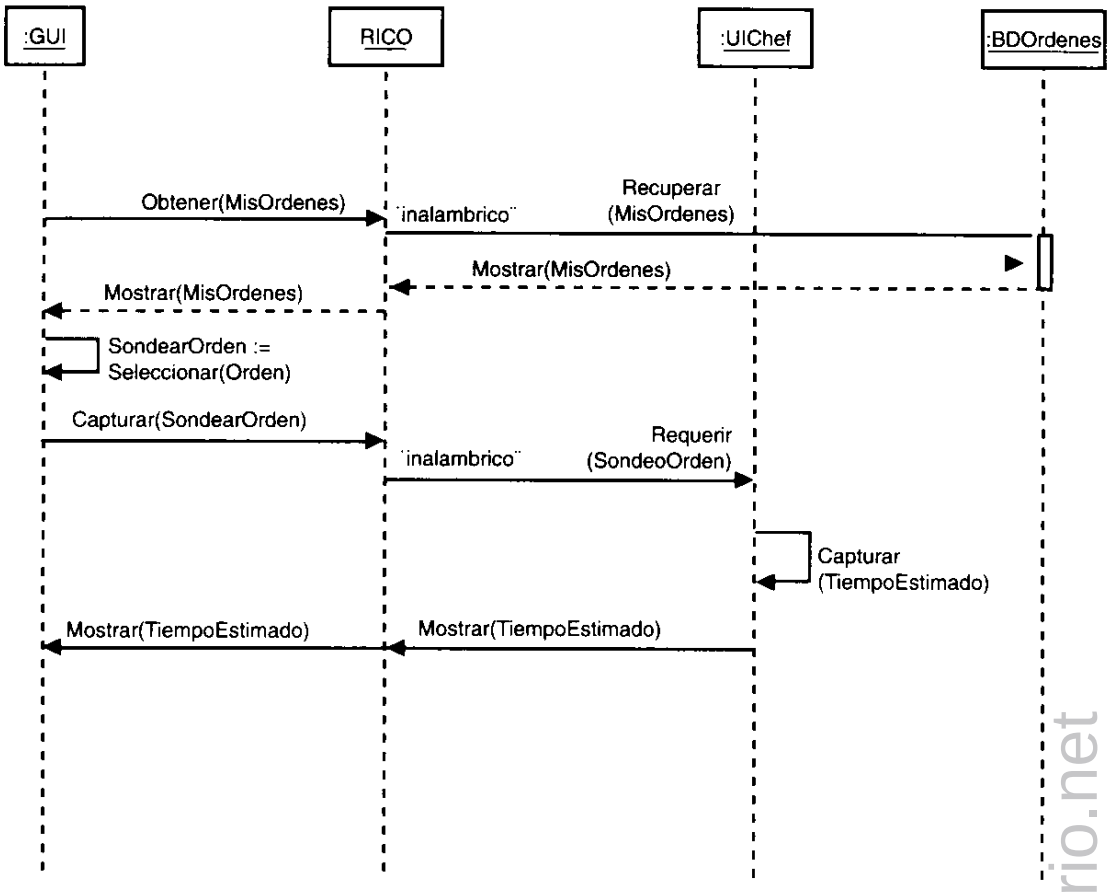
Para continuar con las modificaciones que hemos hecho, posiblemente queramos cambiar el paso 5 a "El mensaje llega a la interfaz del cocinero en la PC de la cocina". También podríamos hacer una entrevista a un chef o dos para preguntarles cómo calculan el tiempo estimado del paso 7. Quizá pudieramos desarrollar un paquete de software que ayudara con ello.

La figura 20.3 hace los honores a este caso de uso.

www.elsolucionario.net

**FIGURA 20.3**

*El diagrama de secuencias de "Sondear el progreso de una orden".*




# Implicaciones

Al ver todos los resultados obtenidos hasta ahora, los señores LaHudra, Nar y Goniff están eufóricos.

"Esto cambiará todo el entorno del negocio de los restaurantes", dijo Nar.

"Sé que hemos logrado algo", dijo LaHudra, "pero, ¿qué quieres decir con 'cambiar todo el entorno del negocio de los restaurantes'?"

"Sí, ¿qué quisiste decir?", preguntó Goniff.

"Bueno, si lo analizan", respondió Nar, "todo el trabajo de un mesero cambiará, igual que el del chef. Los meseros no tendrán que correr de aquí para allá como ahora lo hacen. Siempre estarán al alcance de los clientes dado que siempre se encontrarán en sus áreas de servicio designadas. Irán a la cocina y al bar sólo cuando tengan que hacerlo. Mediante sus palmtops, se convertirán en supervisores del proceso de preparación de las órdenes y administradores de sus áreas. Serán más parecidos a salvavidas que a meseros tradicionales. De hecho, podrán sentarse ocasionalmente mientras se encuentren en sus áreas, dado que el 'trabajo' ya no involucrará andar a las carreras de forma despiadada."

"¿Y qué con los chefs?"

www.elsolucionario.net

"También serán más administrativos. Utilizarán sus equipos de cómputo para asignar órdenes a sus asistentes, y coordinarán lo que proceda en una cocina. Esto será estupendo para grandes cocinas y restaurantes, ahora que lo que hacemos es mover información y no a la gente."

"Hum... Suena bien", dijo LaHudra. "Aparentemente, cuanta más información muevas, moverás menos a la gente y lograrás más. No está mal."

"No del todo", dijo Goniff, ya maquinando la siguiente expansión del negocio.

# Resumen

Luego del análisis del caso de uso, un equipo de desarrollo volvió su atención a los componentes del sistema que sugirieron los casos de uso. ¿Qué son? ¿Cómo interactúan? Esta hora mostró cómo responder a estas preguntas dentro del contexto del desarrollo del sistema RICO.

El objetivo de esto es dar información a los programadores que les facilite su trabajo. Los resultados de este análisis deberán facilitar a los programadores la codificación de los objetos del sistema y la forma en que se comunican entre sí.

Luego de modelar la cooperación entre componentes, el sistema ya está más cercano a convertirse en realidad. Conforme modele la cooperación entre los componentes, podría encontrarse con que será adecuado modificar los casos de uso.

# Preguntas y respuestas

**P Ha mostrado diversas modificaciones a los casos de uso. ¿En realidad así ocurre en un proyecto?**

**R** Indudablemente que sí. Ciertamente, los ejemplos podrían parecer algo amañados; por ejemplo: en realidad posiblemente podríamos haber sabido de la base de datos en el primer caso de uso antes de llegar hasta este punto. Sin embargo, la meta es mostrarle que conforme se desarrolle nuestra noción, el modelo también se desarrollará.

**P ¿Por qué podría ser que los casos de uso originales no capturaran todos los aspectos en primera instancia?**

**R** Porque reflejan los resultados de las sesiones JAD con los usuarios del sistema, no con los desarrolladores de sistemas. Verá que todas las adiciones y cambios estaban relacionados con el sistema, no con el negocio. Una vez que acabe las sesiones con los usuarios potenciales y que tenga la oportunidad de analizar los casos de uso, será común que salgan a la luz modificaciones como esas.

www.elsolucionario.net

www.elsolucionario.net

# Taller

Aquí será donde tendrá la oportunidad de practicar con el modelado de la cooperación entre los componentes de un sistema. Interactúe con el Apéndice A, "Respuestas a los cuestionarios", para encontrar las respuestas. Tal vez necesite utilizar los componentes listados en esta hora para que le ayuden a continuar por encima y más allá de los ejercicios listados y hacer otros diagramas de secuencias y colaboraciones.

## Cuestionario

1. ¿Cómo representaría a un objeto que se genera durante el curso de un diagrama de secuencias?
2. ¿Cómo se representa el tiempo en un diagrama de secuencias?
3. ¿Qué es la "línea de vida"?
4. En un diagrama de secuencias, ¿cómo muestra una "activación" y qué es lo que representa?

## Ejercicios

1. Desarrolle un diagrama de colaboraciones equivalente al de secuencias para el caso de uso del Mesero "Tomar una orden".
2. Cree un diagrama de secuencias para el caso de uso "Tomar una orden de bebida".
3. Seleccione al menos un caso de uso del paquete Chef y desarrolle un diagrama de secuencias. Utilice la lista de componentes mencionados en esta hora. ¿Se necesitan algunos otros?
4. Válgase de su imaginación con ésta: Los casos de uso del paquete Encargado Del Guardarropa parecen ser muy sencillos. ¿Podría adornar a cada uno mediante la adición de uno o dos pasos? ¿Podrían ayudar algunos otros componentes adicionales? Dibuje un diagrama de secuencias para uno de estos casos de uso.
5. Continúe con su proyecto Biblioteca para listar las partes funcionales de su sistema y agregar la colaboración entre objetos. Genere los diagramas de secuencias para los casos de uso resultantes, una vez que haya hecho las modificaciones pertinentes a cada caso de uso.

www.elsolucionario.net

www.elsolucionario.net

# HORA 21

# Diseño del aspecto, sensación y distribución

Ahora nos centraremos en dos importantes aspectos del diseño de sistemas: la interfaz del usuario y la distribución del sistema.

En esta hora se tratarán los siguientes temas:

- Algunos principios generales del diseño de interfaces gráficas
- La sesión JAD de la GUI
- De los casos de uso a las interfaces de usuario
- Diagramas UML para el diseño de la GUI
- Esbozos de la distribución del sistema

Ya ha cumplido con éxito gran parte del análisis conducido por casos de uso. En esta hora, verá dos importantes aspectos del diseño de sistemas. Ambos pueden, a final de cuentas, orientarse a casos de uso, y son muy importantes para el producto final. Las GUI (interfaces gráficas de usuario) determinan qué tan práctico es un sistema. La distribución convierte a la arquitectura física planeada del sistema en una realidad.

www.elsolucionario.net

www.elsolucionario.net

# Algunos principios generales en el diseño de las GUI

El diseño de las GUI, que conjuga al arte y la ciencia, dibuja, bajo la visión del artista gráfico, las conclusiones del investigador de factores humanos y las intuiciones del usuario potencial. Luego de mucha experiencia con interfaces WIMP (Ventanas, Iconos, Menús y Dispositivos apuntadores), han salido a la luz diversos principios generales. He aquí los principales:

1. Entienda lo que el usuario tiene que hacer. Por lo general, los diseñadores de interfaces realizan un *análisis de tareas* para comprender la naturaleza del trabajo del usuario. Nuestro análisis de casos de uso corresponde más o menos a esto.
2. Haga que el usuario sienta que tiene el control de la interacción. Siempre incluya la posibilidad de que el usuario cancele una acción luego que la haya iniciado.
3. Dé al usuario diversas formas de realizar cada acción relacionada con la interfaz (como el cierre de una ventana o archivo) y disculpe con elegancia los errores del usuario.
4. Dadas nuestras influencias culturales, nuestros ojos se dirigen a la esquina superior izquierda de la pantalla. Coloque la información de mayor prioridad allí.
5. Aproveche las relaciones de espacio. Los componentes de la pantalla que estén relacionados deberán aparecer uno junto de otro, quizá con un marco que los bordee.
6. Destaque la legibilidad y la comprensión (¡palabras con las que tenemos que vivir!). Utilice la voz activa para comunicar ideas y conceptos.
7. Aunque tal vez tenga la capacidad de incluir miles de colores en la pantalla, restrinja los que vaya a utilizar. De hecho, haga una fuerte limitación de ellos. Demasiados colores distraerán al usuario de la tarea por realizar. Por cierto, se recomienda dar al usuario la opción de modificar los colores.
8. Si piensa en utilizar colores para indicar algo, recuerde que no siempre será fácil para el usuario asociar un color con un significado. A su vez, tenga en cuenta que ciertos usuarios (cerca del 10% de los adultos masculinos) sufren de daltonismo, y se les podría dificultar distinguir a un color de otro.
9. Como con el color, restrinja el uso de fuentes. Evite las fuentes cursivas y ornamentales. "Haettenschweiler" es un nombre de fuente que puede ser divertido pronunciar, pero que no fomenta una facilidad de uso.
10. Intente mantener, tanto como sea posible, los componentes (como los botones y cuadros de lista) del mismo tamaño. Si utiliza componentes de tamaños dispares, múltiples colores y fuentes, creará una terrible interfaz que los especialistas en el área conocen como un diseño de "pantalón de payaso".
11. Alinee los componentes y campos de datos a la izquierda: alíneelos de acuerdo con sus bordes izquierdos. Esto reduce los movimientos oculares cuando el usuario tiene que revisar la pantalla.

www.elsolucionario.net

www.elsolucionario.net

12. Cuando el usuario tiene que leer y procesar información y luego hacer clic en un botón, coloque los botones en una columna a la derecha de la información, o en una fila debajo y a la derecha de la misma. Esto es conveniente debido a la tendencia natural (o cultural) de leer de izquierda a derecha. Si uno de los botones es el predeterminado, destáquelo y colóquelo como el primero en el conjunto.

Esta docena de principios no son los únicos, pero nos dan una idea de lo que está involucrado en el diseño de una GUI. El reto es comunicar la información adecuada en un contexto visual elemental, directo e intuitivo.

Para ver mayor información respecto al diseño de interfaces, en especial las que están relacionadas con Windows, visite `http://msdn.microsoft.com/UI`. En esa misma página encontrará información para el diseño de páginas Web o puede visitar `www.useit.com`. En ambos casos, la información se encuentra en inglés.

La figura 21.1 le muestra lo que ocurre cuando pone en acción algunos de estos principios y la figura 21.2 le muestra lo que sucede cuando no lo hace.

**FIGURA 21.1**

*Aplicación de los principios para el diseño de GUI.*

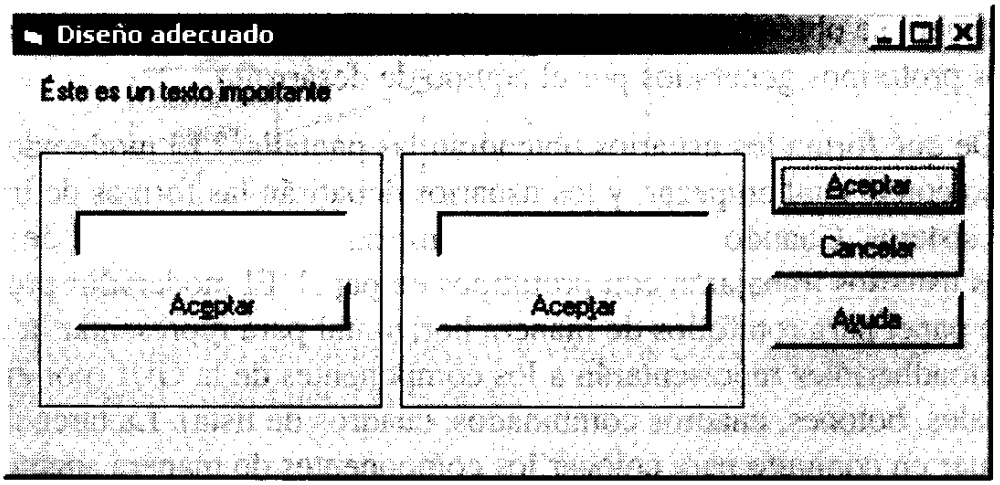
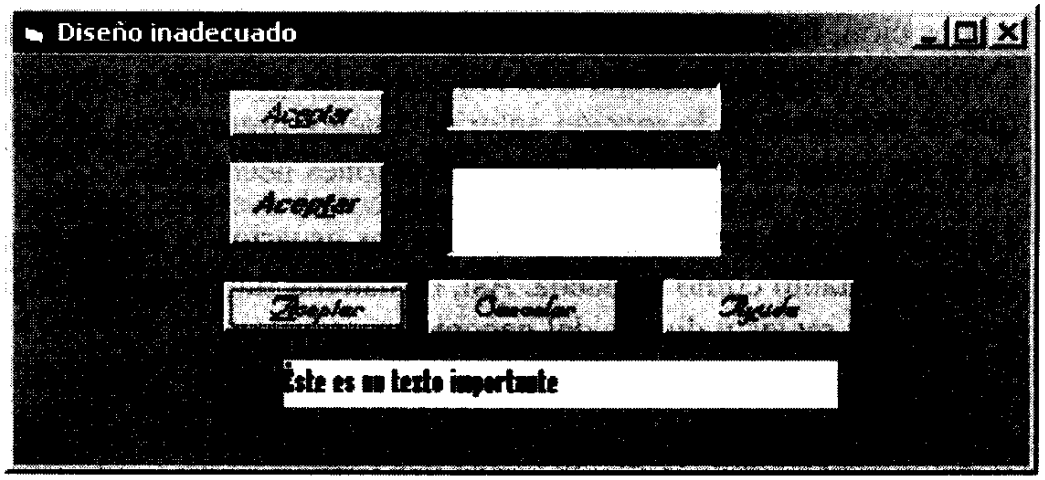



**FIGURA 21.2**

*El resultado de no aplicar los principios para el diseño de GUI.*




www.elsolucionario.net

www.elsolucionario.net

# La sesión JAD para la GUI

Aunque la sesión JAD no tendrá una conexión directa con el UML, es buena idea hablar de la forma en que los usuarios potenciales determinan a la GUI. Nuevamente, se realizará una sesión para el Desarrollo conjunto de aplicaciones (JAD).

Para esta sesión reunirá a los usuarios potenciales del sistema. En el caso de RICO, reuniremos a los meseros, chefs, asistentes de los meseros, asistentes de los chefs, mozos de piso y a los encargados del guardarropa. Los integrantes del equipo de desarrollo que estarán presentes serán los programadores, analistas, modeladores y un moderador. El objetivo será comprender las necesidades de los usuarios e implementar una interfaz de acuerdo con sus ideas (una interfaz que integre sutilmente al sistema con los procesos del negocio). La antigua manera de desarrollar un sistema (por ejemplo, escribir un programa desde sus inicios, moldear el comportamiento del usuario para que puedan interactuar con él y modificar los procesos del negocio para que se adapten a él) ya es obsoleta.

Para mantener la eficiencia de la sesión, programará a los usuarios en grupos de acuerdo con el rol que desempeñen. Planeará la cantidad de tiempo que se llevará cada sesión de acuerdo con la cantidad de casos de usos en el paquete de cada rol. Esto es sólo un vago lineamiento, claro, ya que algunos casos de uso son más complejos que otros. Recuerde, a su vez, que podrían aparecer nuevos casos de uso al diseñar la GUI.

La participación de los usuarios en la sesión es un asunto que envuelve a dos partes. En la primera se obtendrán las pantallas de la interfaz del usuario. En la segunda se aprobarán los prototipos generados por el equipo de desarrollo.

¿De qué forma los usuarios obtendrán las pantallas? El moderador sugiere un caso de uso con el cual empezar, y los usuarios debatirán las formas de implementarlo mediante el sistema. Cuando estén listos para empezar a hablar al nivel de una pantalla específica, los usuarios trabajarán con prototipos en papel. El moderador proporciona una gran hoja de papel para rotafolios de manera horizontal para representar la pantalla. Las notas autoadheribles representarán a los componentes de la GUI (por ejemplo: menús contextuales, botones, cuadros combinados, cuadros de lista). La tarea de los usuarios será trabajar en conjunto para colocar los componentes de manera adecuada.

Cuando lleguen a un acuerdo en los componentes que deban estar en una pantalla y su disposición, los miembros del equipo de desarrollo generarán pantallas de prototipo. Conforme hagan su trabajo, irán utilizando los principios adecuados del diseño de GUIs que se indicaron en la sección anterior. Luego, presentarán tales pantallas en computadoras y los usuarios les harán las modificaciones necesarias.

La meta de todo esto, claro está, es que los usuarios (y no los desarrolladores) conduzcan el proceso tanto como se pueda. Así, el sistema funcionará de manera óptima en las actividades reales y diarias del negocio.

www.elsolucionario.net

www.elsolucionario.net

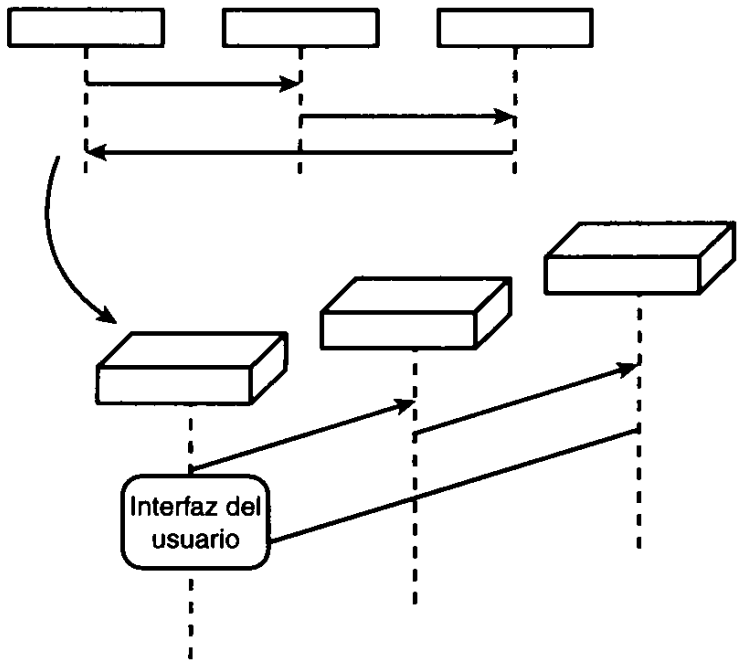
# De los casos de uso a las interfaces de usuario

Los casos de uso describen la forma en que se utiliza el sistema. Por ello, la interfaz del usuario tiene que servir como medio para implementar los casos de uso.

Imagine un diagrama de secuencias de un caso de uso como una visión de un caso de uso. Si pudiéramos "girar" tal visión en tres dimensiones, para que la parte del extremo izquierdo del diagrama de secuencias se desprendiera de la página y quedara frente a nosotros, veríamos la interfaz que llevaría al usuario a la secuencia (vea la figura 21.3).

**FIGURA 21.3**

*El supuesto "giro" del diagrama de secuencias nos pone de manifiesto la interfaz del usuario.*




Examinemos los casos de uso del paquete Mesero y veamos cómo encajan en la interfaz de RICO. Nuevamente, aquí están tales casos de uso:

- Tomar una orden
- Transmitir la orden a la cocina
- Cambiar una orden
- Recibir una notificación de la cocina
- Sondear el progreso de la orden
- Notificar al chef del progreso de los clientes en sus alimentos
- Totalizar una cuenta
- Imprimir una cuenta
- Llamar a un asistente

www.elsolucionario.net

www.elsolucionario.net

- Llamar a un mozo de piso
- Tomar una orden de bebida
- Transmitir una orden de bebida al bar
- Obtener un acuse de recibo
- Recibir una notificación del bar

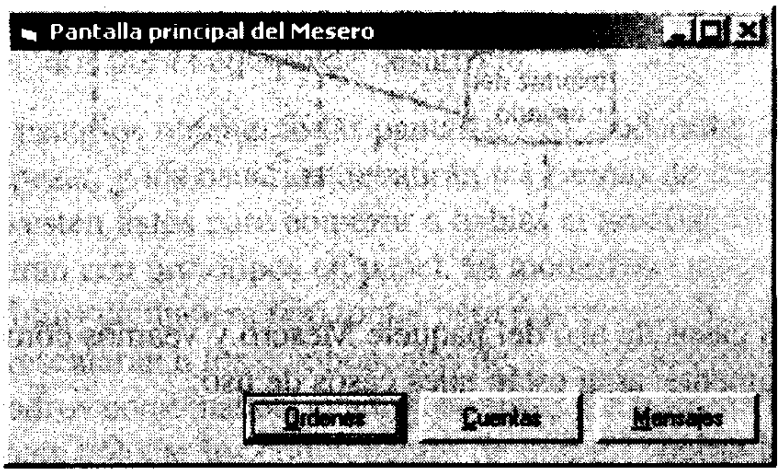
La interfaz del Mesero tiene que incluir a todos estos casos de uso.

Una forma de empezar es dividir al conjunto de casos de uso en grupos. Con tres de ellos será suficiente. Uno englobará a las órdenes ("Tomar una orden", "Cambiar una orden", "Sondear el progreso de la orden", "Tomar una orden de bebida"). Otro grupo englobará a las cuentas ("Totalizar una cuenta", "Imprimir una cuenta"). El tercer grupo englobará al envío y recepción de los mensajes ("Notificar al chef del progreso de los clientes en sus alimentos", "Llamar a un asistente", "Llamar a un mozo de piso", "Transmitir una orden de bebida al bar", "Obtener un acuse de recibo", "Recibir una notificación del bar").

Tal vez necesitemos empezar con una pantalla principal que llevaría al mesero por diversas pantallas hacia todos los demás grupos de casos de uso. Necesitamos tener la posibilidad de navegar de un grupo a cualquier otro. Dentro de un grupo, necesitamos navegar a cualquier caso de uso dentro del grupo. La figura 21.4 muestra una idea inicial de la pantalla principal. Esto tendrá que encontrarse en una palmtop, por lo que tal vez sea necesario ajustar su tamaño de alguna forma.

**FIGURA 21.4**

*Primera idea de una pantalla principal del mesero.*




Nuestra sesión JAD podría llegar a convenir que la navegación dentro de un grupo se realizaría mediante botones localizados a la derecha de la pantalla, en tanto que la navegación entre grupos se realizaría con botones en la parte inferior de la pantalla. La figura 21.5 le muestra una idea básica de una de las interfaces del Mesero: la de los casos de uso relacionados con las órdenes.

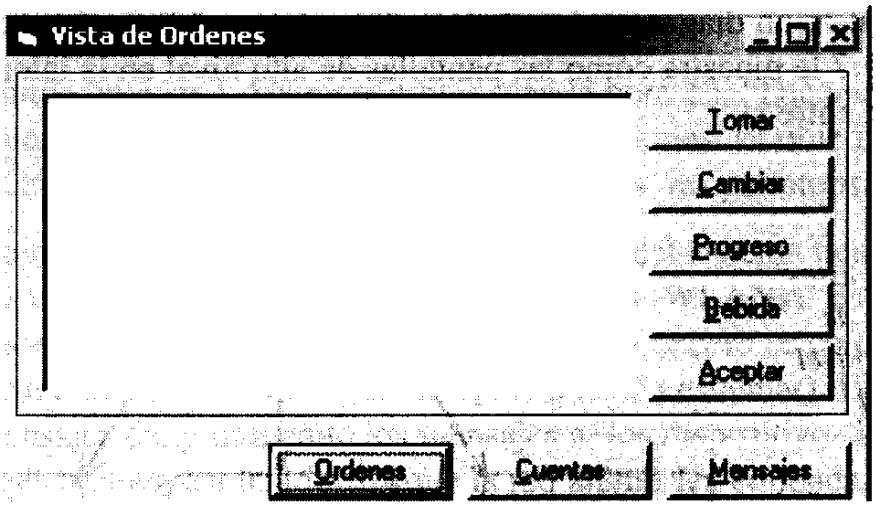
Esta pantalla se abre en el modo Ordenes. El gran cuadro blanco se desplazará con una copia del menú con casillas de verificación que el mesero seleccionará para indicar las selecciones del cliente (cuando veamos la interfaz, tendremos un especial cuidado al utilizar la palabra "menú"). Al hacer clic en Aceptar se genera la orden y se envía a la PC de la cocina. Al hacer clic en un botón de la derecha se obtendrán sus características asociadas.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 21.5**

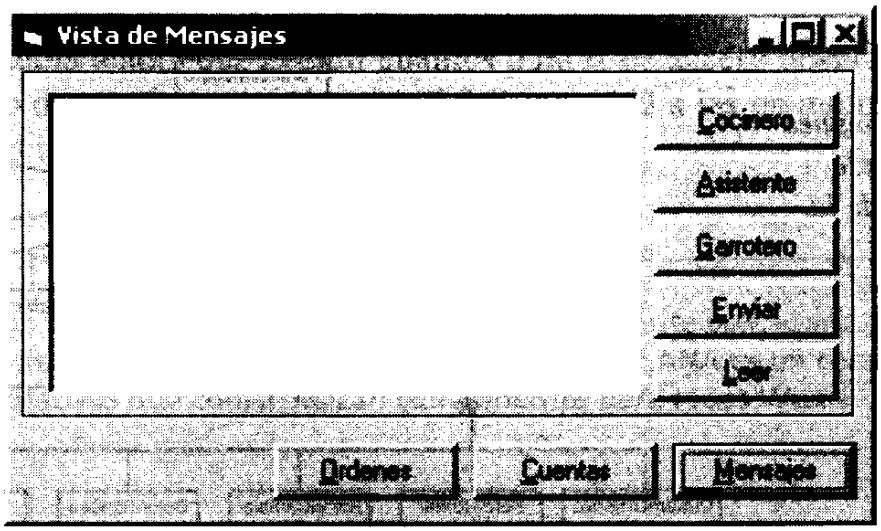
*Pantalla para los casos de uso relacionados con las órdenes.*




Cuando se haga clic en los botones de la fila inferior se obtendrá otro grupo de capacidades. Por ejemplo, el botón Mensaje traerá la pantalla de la figura 21.6. Por cierto la interfaz de usuario no debe ser sólo visual. Esta interfaz tiene que incorporar algún elemento audible para notificar a un mesero que le ha llegado un mensaje. Así, hará clic en el botón Leer para que se le presente una lista desplegable de mensajes.

**FIGURA 21.6**

*Pantalla para los casos de uso relacionados con los mensajes.*




www.elsolucionario.net

# Diagramas UML para el diseño de la GUI

El UML no hace recomendaciones específicas respecto a los diagramas para diseños GUI. No obstante, en una hora anterior indicamos una posibilidad: recuerde que en la hora 8, "Diagramas de estados", presenté un ejemplo que trataba de los cambios de estado en una GUI. Aunque tal ejemplo profundizó en la funcionalidad de una GUI más de lo que deberíamos en este punto, sugiere que los diagramas de estados son útiles cuando se trata de interfaces de usuario.

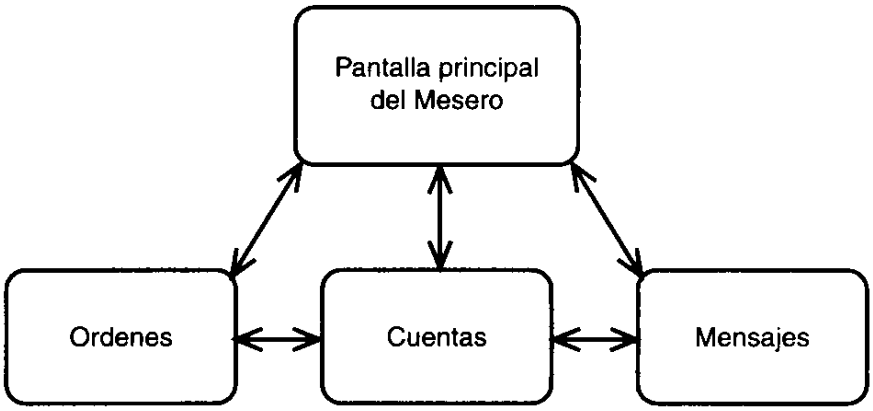
En la hora 24, "El futuro del UML", presentaremos algunas ideas de cómo extender el UML para modelar las GUIs.

www.elsolucionario.net

Debería usar un diagrama de estados para mostrar el flujo de una interfaz de usuario. La figura 21.7 le muestra cómo las pantallas de alto nivel en la interfaz del Mesero se conectan entre sí.

**FIGURA 21.7**

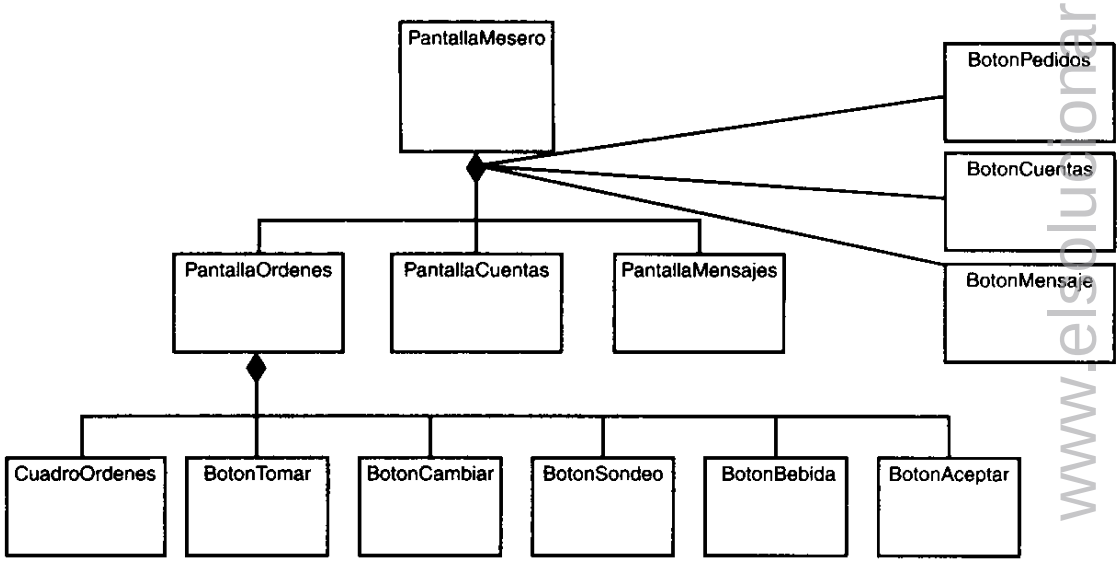
*Un diagrama de estados para el flujo de pantalla de alto nivel en la interfaz del Mesero.*




Como una pantalla en particular consta de varios componentes, un diagrama de clases para un objeto compuesto es adecuado para modelar una pantalla. La figura 21.8 muestra un diagrama para un objeto compuesto que corresponde con la pantalla de la figura 21.5.

**FIGURA 21.8**

*Un diagrama compuesto que corresponde con la pantalla de la figura 21.5.*




www.elsolucionario.net

# Esbozos de la distribución del sistema

Luego de que el segmento de análisis GRAPPLE ha producido el concepto general del sistema RICO, un ingeniero de sistemas empezará a pensar cómo deberá lucir la arquitectura física. Empezará a considerar las alternativas en topologías de red y la forma de implementarlas de manera inalámbrica, y comenzará a resolver qué componentes de software pertenecerán a determinados nodos en la red. Este segmento de diseño no tiene que esperar a que el análisis se haya completado. Sus acciones pueden proceder de forma paralela con las de los segmentos GRAPPLE, como con el diseño de la GUI.

La meta se centra en que el administrador del proyecto sondee todas las acciones en todos los segmentos.

www.elsolucionario.net

## La red

Al recordar los distintos tipos de LAN disponibles (vea la hora 13, "Diagramas de distribución"), el ingeniero del sistema cuenta con diversas opciones. El objetivo es elegir aquél que se integre mejor con la conectividad inalámbrica en las palmtops.

Para comprender algunas de las decisiones que el ingeniero del sistema tiene que tomar, vamos a profundizar un poco en las redes LAN inalámbricas (WLANs). Es frecuente el caso de que un transceptor de radio conocido como *punto de acceso* se coloque en un lugar fijo y se conecte a una LAN (por algún medio alámbrico estándar). El punto de acceso recibe los mensajes de, y transmite los mensajes a, los dispositivos inalámbricos, y cambia los mensajes recibidos a una forma que la red alámbrica entienda. Varios puntos de acceso aumentarán el rango de la WLAN y la cantidad de usuarios que podrán acceder a ella.

¿De qué manera los usuarios accederían a la WLAN? En RICO, harán la conexión mediante adaptadores LAN inalámbricos integrados con sus palmtops. No importa cuántos puntos de acceso se incorporen en la red, cada palmtop se asocia con sólo un punto de acceso y su área de cobertura, lo que se conoce como *microcelda* (que es similar a una celda que funciona en los teléfonos celulares).

El ingeniero del sistema tendrá que decidir cuántos puntos de acceso deberá tener el restaurante, qué tipo de adaptador inalámbrico LAN integrará a las palmtops, y el tipo y disposición de la red alámbrica.

Si algo de esto le ha interesado para su WLAN, visite `www.wlana.com`, el sitio Web de la Alianza inalámbrica LAN (WLANA). WLANA es un consorcio de corporativos que comercializan componentes para redes WLAN.

Vamos a suponer que el ingeniero del sistema decide utilizar una red thin ethernet para la LAN (vea la hora 13).

## Los nodos y el diagrama de distribución

Ya enumeramos los nodos en nuestro sistema. Los meseros, asistentes de meseros y mozos de piso tendrán palmtops. La cocina, guardarropa y bar tendrán computadoras de escritorio. Cada computadora se conectará a una impresora. Además, cada área de servicio tendrá una computadora de escritorio conectada a una impresora para que pueda imprimir cuentas y obtenerlas sin caminar demasiado (un servidor de impresión, si se puede llamar así).

www.elsolucionario.net

www.elsolucionario.net

No obstante, en este punto veremos un ejemplo primordial del porqué es mejor empezar a pensar en las características de distribución desde el principio. Conforme se avanza, el ingeniero del sistema y el equipo de desarrollo tendrán que tomar una decisión de negocios: las palmtops requieren adaptadores LAN inalámbricos, que podrían ser muy costosos, y el software asociado podría estar fuera de las necesidades iniciales. Una posibilidad potencialmente menos costosa es la de cambiar los dispositivos móviles de palmtops a otras conocidas como Handheld que ejecuten Windows CE y utilicen una tarjeta PC para conectarse de manera inalámbrica a la LAN. Una característica importante de esta alternativa es que los programadores pueden utilizar herramientas de desarrollo ya conocidas para crear el software y llevarlo a las computadoras móviles.

¿Por qué no utilizar una palmtop con Windows CE? Cuando escribí estas líneas, los dispositivos Palmtop con Windows CE no tenían ranuras para tarjetas PC, y la única tarjeta PC WLAN para Windows CE era la Proxim RANGELAN2. Conclusión: resuelva estos detalles en los procesos iniciales del proyecto.

En cualquier caso, asumamos que el equipo de desarrollo y los usuarios deciden utilizar la solución Windows CE/Proxim, con la idea de que el desarrollo será más eficiente y económico si los desarrolladores pueden aprovechar sus herramientas de desarrollo para crear el código necesario. Como comentario al margen, de acuerdo con la velocidad a la que avanzan las cosas, podría aparecer en cualquier momento un nuevo dispositivo Palmtop con Windows CE y una ranura para tarjetas PC.

Esta línea de razonamiento orienta al ingeniero del sistema hacia una solución Proxim para un punto de acceso a la LAN; al igual que la tarjeta PC, también se llama RANGELAN2.

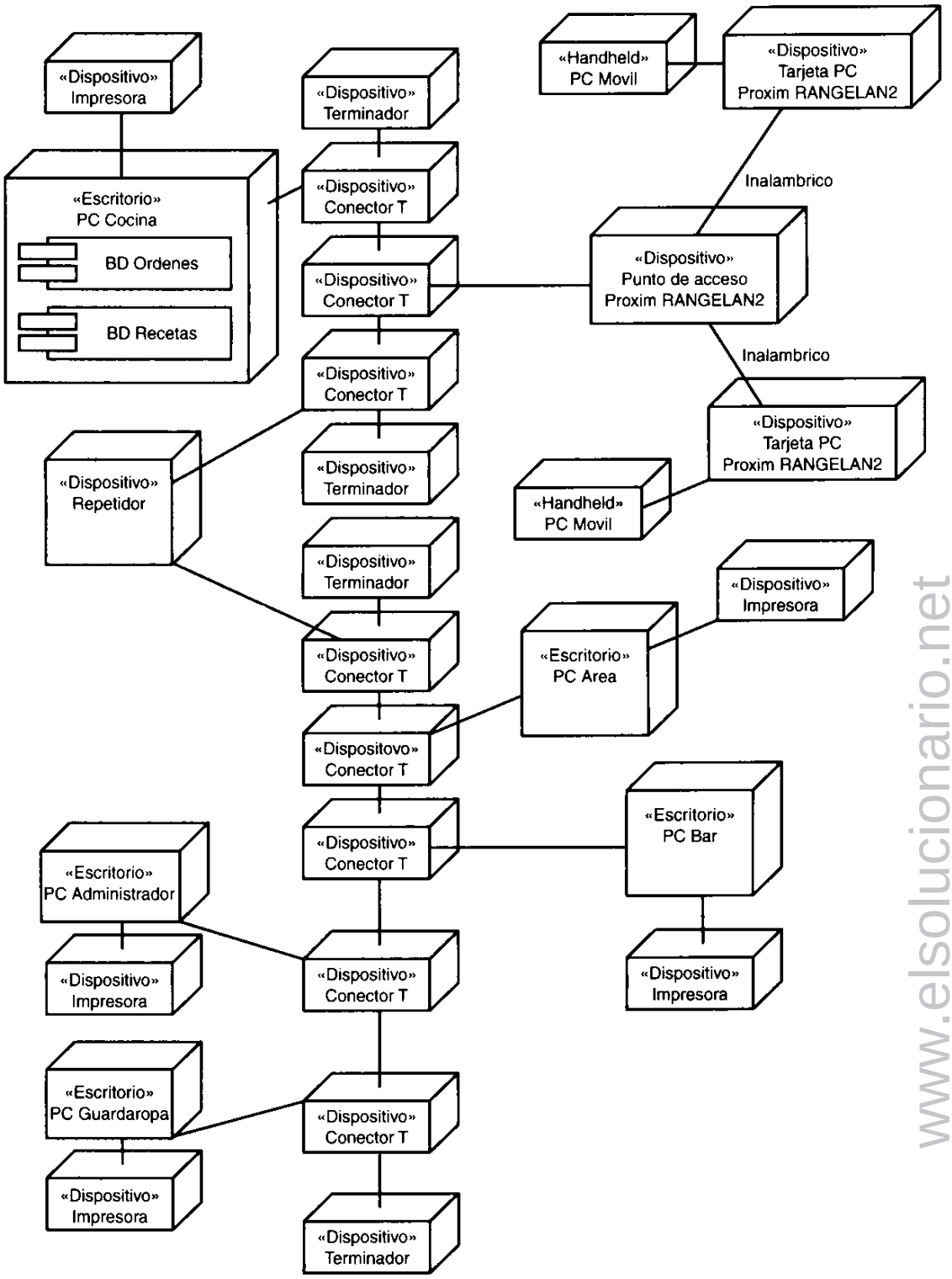
Para ilustrar la distribución, el ingeniero del sistema genera el diagrama de distribución que se muestra en la figura 21.9.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 21.9**

*Diagrama de distribución para RICO.*




# Siguientes pasos

El equipo de desarrollo ha ido de los casos de usos a las interfaces y a las WLANs. ¿Qué sigue?

Para empezar, los analistas pondrán en orden al modelo. Verán el diccionario de modelos y pondrán en orden cualquier ambigüedad. Se asegurarán de que toda la terminología sea utilizada de manera consistente en todos los diagramas y que los problemas con términos

www.elsolucionario.net

www.elsolucionario.net

como "menú" no propendan a confusiones. Cuando se completen todas las partes del análisis y diseño de GRAPPLE, el equipo compilará los resultados en un documento de diseño y enviará copias al cliente y a los programadores.

A los programadores o desarrolladores es a quienes corresponde convertir el diseño en código (la parte de la codificación va más allá de los alcances de este libro).

El código será probado, vuelto a escribir de acuerdo con los resultados de las pruebas, y vuelto a probar (proceso que continuará una y otra vez hasta que el código pase todas las pruebas). El análisis de los casos de uso conforma los fundamentos de las pruebas.

Los especialistas en la documentación empezarán a crear la información del sistema, y crearán también los materiales de entrenamiento. Un buen proyecto para la creación de la documentación debería proceder como uno para el desarrollo de un sistema, con una cuidadosa planeación, análisis y pruebas, y deberá comenzar en las primeras instancias del proceso de desarrollo.

Con un documento de diseño bien organizado e informativo y con un análisis y diseño sólidos, los siguientes pasos deberían proceder sin problemas durante todo el proceso de desarrollo.

La idea principal es enfocar los esfuerzos primordialmente en el análisis y diseño para que la implementación tenga muy pocos traspiés y que el proyecto dé por resultado un sistema que cumpla por completo con las necesidades del cliente.

# ...Y ahora, unas palabras de nuestros patrocinadores

Los señores LaHudra, Nar y Goniff no podrían estar más emocionados por la forma en que ha transcurrido el proyecto de desarrollo. El equipo de desarrollo los ha mantenido informados del proceso y les ha dado diseños basados en UML que les muestran el punto en que se encuentra el proyecto. Incluso están satisfechos por la forma en que el equipo solucionó el gran problema de distribución respecto a los dispositivos móviles.

Todo el esfuerzo ha despertado sus imaginaciones y los ha impulsado a buscar nuevas formas de utilizar la tecnología (tanto dentro como fuera del mundo restaurantero). Han caído en la cuenta que la mayoría de los procesos de negocios involucran la transmisión de la información. El grado en que la tecnología acelera tal transmisión, representa una enorme ventaja competitiva.

## Mejorar el trabajo de la fuerza de ventas

Fuera del mundo restaurantero, los tres empresarios ven el potencial de volver a utilizar las ideas de la LAN inalámbrica para una fuerza de ventas móvil dentro de un enorme área de trabajo. El volverlas a emplear no debería ser difícil, ya que toda la información de los modelos está intacta.

www.elsolucionario.net

www.elsolucionario.net

Una aplicación de esta idea podría ser en las enormes tiendas departamentales o de autoservicio. Los vendedores de piso de dichas tiendas podrían beneficiarse de un dispositivo de mano que acceda a la información de los productos mediante una LAN inalámbrica. Un sistema como éste podría ayudar al vendedor a responder preguntas respecto al lugar donde se encuentra el producto en la tienda, si hay en existencia y cómo se podría utilizar.

Esto tiene algunas implicaciones intrigantes tanto para el vendedor como para el cliente. Los clientes estarían seguros que siempre obtendrán la información más reciente y concisa del vendedor. Un nuevo vendedor entrenado sobre cómo utilizar el sistema podría, con rapidez, empezar a trabajar aunque tuviera una mínima capacitación sobre el almacén.

LaHudra, Nar y Goniff pronto invadirán el mundo de las mejoras al hogar.

## Expansiones en el mundo restaurantero

Esta idea de la fuerza de ventas móvil no es suficiente para LaHudra, Nar y Goniff. No quieren hacer nada que no incluya a la tecnología para revolucionar los negocios restauranteros. Creen que podrían crear restaurantes basados en RICO en las principales ciudades del mundo. Creen que la tecnología simplificará la sensación de comer fuera de casa y lo hará más conveniente para que todos salgan a hacerlo.

Goniff, como siempre en busca de nuevas formas de ganar dinero, tiene un buen rato pensado en esto (¡al menos desde que finalizó la hora 20!).

"Amigos," les dijo a sus socios, "si construimos restaurantes en todas las ciudades principales, podríamos llevar la tecnología a un siguiente paso y transmitir la información por doquier."

"¿Cómo?" preguntó Nar, que siempre ha sido algo torpe.

"Piénsenlo. Si nos internacionalizamos, podríamos entrar en la Web y..."

LaHudra interrumpe: "Un momento. Ya estamos en la Web. Hay muchos que visitan www.lahudranargoniff.com, ¿O no?"

"Déjame terminar, LaHudra. Podríamos utilizar la Web para que la gente 'vaya' a todos estos restaurantes. Utilizaríamos la Web para darles un emparedado gratuito."

"¡Qué?" preguntaron Nar y LaHudra al unísono y con incredulidad.

"Vamos a ver. Orientamos una página de nuestro sitio Web a nuestra división de restaurantes. Alguien entra a la página, da su nombre y otra información, y se le pide que elija el emparedado que desee. Si nuestra base de datos muestra que el visitante no ha hecho esto con anterioridad, se le presentaría otra página donde podrá imprimir un cupón para un emparedado. Llevará el cupón al restaurante más cercano. Obtendrá el emparedado, lo comerá, le gustará y regresará como un cliente normal."

www.elsolucionario.net

www.elsolucionario.net

"Sí, pero la Web puede verse en cualquier parte", dijo Nar. "Supón que alguien no vive cerca de uno de nuestros restaurantes y, de todas maneras, desea el emparedado."

"¡Aguarden! ¡Ya sé!", dijo LaHudra. "Pueden utilizar su tarjeta de crédito para pagar un cargo nominal por envío en el sitio Web, y nuestro restaurante más cercano se lo enviará a su hogar en un frío y económico contenedor. Podrán colocar el emparedado en su horno de microondas y calentarlo. Así, podrán comer un producto de LaHudra, Nar y Goniff donde estén. Luego, cuando viajen a una ciudad donde haya uno de nuestros restaurantes, querrán comer allí."

"Por cierto, ¿para qué sería el resto de la información que ellos capturen antes de imprimir el cupón?", preguntó Nar.

"Te lo diré," dijo Goniff. "Utilizaremos esta indagación para enviarles por correo electrónico información promocional de nuestros demás negocios, de acuerdo con su demografía (siempre y cuando indiquen que quieren recibirla).

"¿Dónde está el equipo de desarrollo? Tenemos mucho trabajo qué hacer."

# Resumen

Cuando su proyecto se encuentre en el segmento de diseño, habrá dos elementos por enfocar que serán la interfaz del usuario y la distribución del sistema. Ambos son finalmente conducidos por los casos de uso y de extrema importancia.

El diseño de la interfaz del usuario depende de una visión artística y de una investigación científica. Varios de los principios del diseño de la interfaz han salido a la luz luego de años de trabajo con interfaces WIMP. Esta hora le mostró algunos de ellos. Téngalos en cuenta cuando su equipo de desarrollo diseñe GUIs.

Los casos de uso conducen el diseño de la interfaz del usuario. El sistema tiene que permitir al usuario completar cada caso de uso, y la interfaz es la puerta de acceso hacia cada uno de ellos.

De forma simultánea con varios de los procesos del proyecto, el ingeniero de sistemas del equipo se orientará a la arquitectura física. La arquitectura está conducida por los casos de uso dado que el uso del sistema finalmente determina la naturaleza física y la disposición del mismo. El ingeniero de sistema otorga un diagrama de distribución UML que muestra los nodos, los componentes de software que haya en cada nodo y las conexiones entre nodos. Aunque los detalles de la distribución aparecen en las etapas avanzadas del proceso GRAPPLE, no hay razón para no empezar a pensar en ellos en etapas previas. Como se mostró en esta hora, pueden aparecer detalles fundamentales que necesitarán ser resueltos.

www.elsolucionario.net

www.elsolucionario.net

Luego que el sistema ha sido modelado, la información del modelado puede volver a utilizarse en diversos contextos. El modelo puede impulsar miles de ideas nuevas para negocios.

# Preguntas y respuestas

**P Una vez que los usuarios hayan generado un prototipo en papel, ¿en realidad será necesario crear la pantalla y mostrárselas? Después de todo, ya generaron la pantalla en el papel y colocaron adecuadamente los componentes. ¿No podrían esperar y ver las pantallas en el sistema una vez que esté finalizado?**

**R** Es indudable que tendrá que mostrarles a los usuarios una pantalla verdadera ("verdadera" en el sentido de la computadora). Para empezar, es muy probable que los usuarios vean cosas en la pantalla que no vieron en el papel. Otra razón (relacionada con la primera) es que las dimensiones de las notas autoadheribles sólo se aproximan a las de los componentes en la pantalla. Colocar las notas autoadheribles podría distorsionar la relación de espacio entre los componentes de la pantalla. La pantalla podría lucir un tanto distinta al prototipo en papel. A su vez, las capturas de la pantalla se convertirán en partes muy valiosas para su documento de diseño.

**P Sé que esto no está directamente relacionado con el UML, pero uno de los principios de la GUI que mencionó fue dar al usuario diversos medios para realizar las acciones relacionadas con la interfaz. ¿Por qué esto es importante?**

**R** Lo es porque no podrá predecir todos los contextos en donde un usuario realizará una acción. En ocasiones el usuario utilizará una aplicación que utiliza mucho el teclado, y una combinación de teclazos será más adecuada que un clic con el ratón. En ocasiones el usuario utilizará mucho el ratón, y un clic será más adecuado. Dar ambos medios para realizar lo mismo hace que la interacción sea más fácil para el usuario.

**P Y hablando de preguntas que no se relacionan con el UML, quisiera saber el porqué del principio de la "voz activa" en la GUI.**

**R** Los estudios demuestran que las personas comprenden mejor la voz activa que la pasiva. A su vez, la voz activa requiere, por lo general, menos palabras y, con ello, ocupará menos del precioso espacio de la pantalla que la voz pasiva. Los usuarios (así como los editores y jefes de redacción) aprecian si sus instrucciones dicen: "Haga clic en el botón Siguiente para continuar" en lugar de "El botón Siguiente debería ser oprimido por usted para que continúe el proceso".

www.elsolucionario.net

www.elsolucionario.net

# Taller

Este taller verifica su conocimiento de las cuestiones relacionadas con el diseño del aspecto y sensación de un sistema, y para orientarse a la arquitectura física correspondiente. Diseñe bien sus respuestas y, luego, vaya al Apéndice A, "Respuestas a los cuestionarios" para verificarlas.

## Cuestionario

1. ¿Qué es un análisis de tareas?
2. ¿Cuál análisis de los que ya hemos hecho es un vago equivalente de un análisis de tareas?
3. ¿Qué se entiende por un diseño de tipo "pantalón de payaso"?
4. Dé tres razones para restringir el uso del color en una GUI.

## Ejercicios

1. Utilice un diagrama de estados UML para modelar la interfaz del usuario del chef.
2. Utilice papel y lápiz para diseñar al menos una de las pantallas de la interfaz del usuario del chef. Empiece por agrupar los casos de uso y, luego, cíñase a las convenciones de la sesión JAD. Si puede utilizar Microsoft Visual Basic, o alguna otra herramienta para el diseño visual de pantallas, intente utilizarla para completar este ejercicio.
3. Nuevamente, continúe con el análisis de las tareas que haya obtenido en su proyecto de la Biblioteca. No olvide hacer un análisis de la distribución.

www.elsolucionario.net

www.elsolucionario.net

# HORA 22

# Noción de los patrones de diseño

Ahora que ha comprendido los fundamentos del UML y ha visto cómo utilizarlo en el contexto de un proyecto de desarrollo, finalizaremos la parte II con una semblanza de la aplicación del UML a un área novedosa: los patrones de diseño.

En esta hora se tratarán los siguientes temas:

- Parametrización
- Patrones de diseño
- Cadena de responsabilidad
- Uso de nuestros propios patrones de diseño
- Ventajas de los patrones de diseño

En las 21 horas anteriores ha visto muy diversos temas. Desde los diagramas de clases hasta los de secuencias, desde los diagramas de estados hasta las sesiones JAD, la meta fue prepararlo para aplicar el UML en situaciones que ocurren frecuentemente en el mundo real.

www.elsolucionario.net

www.elsolucionario.net

Ahora, cambiaremos nuestro rumbo un poco. En esta hora exploraremos una de las aplicaciones del UML que, posiblemente, se harán más populares. Esta aplicación, la representación de patrones de diseño, captura la esencia de las soluciones que han funcionado una y otra vez en proyectos y situaciones reales.

# Parametrización

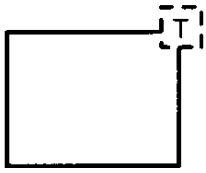
En la hora 2, "Orientación a objetos", mencioné que una clase es una plantilla para la generación de objetos. Le dije que podía concebir a una clase como un molde de galletas generador de nuevos objetos. Un objeto, recordará, es una instancia de una clase.

Para ayudarle a recordar, veamos de nuevo el ejemplo de nuestra lavadora. La especificación de la clase lavadora (o, para utilizar una notación correcta, la clase Lavadora) incluía los atributos marca, modelo, numeroDeSerie y capacidad, y las operaciones agregarRopa(), agregarDetergente y quitarRopa(), con lo que tendríamos una forma de generar nuevos objetos provenientes de nuestra clase Lavadora. Cada vez que queramos crear un objeto, asignaremos valores a los atributos.

El propio UML le permite ir un paso más allá. Le da un mecanismo para crear clases de una forma similar a la creación de objetos. Podrá establecer una clase de manera que cuando asigne valores a un subconjunto de sus atributos habrá creado una clase en lugar de un objeto. A este tipo de clase se le conoce como clase *parametrizada*. Su representación en el UML aparece en la figura 22.1. El cuadro punteado del extremo superior derecho contiene los parámetros a los que asignará valores para generar la clase. A estos se les llaman *parámetros desvinculados*. Cuando les asigne valores, los vinculará con ellos. La T del cuadro punteado es un clasificador que indica que la clase es una plantilla para crear otras clases.

**FIGURA 22.1**

*El icono UML para una clase parametrizada.*




He aquí un ejemplo. Suponga que establece un SerVivo como clase parametrizada. Los parámetros desvinculados podrían ser géneros y especies, junto con los atributos estándar de nombre, estatura y peso, como se muestra en la figura 22.2.

Si vincula a género con "homo" y a especie con "sapiens", creará una clase llamada "Humano". El nombre de la clase se vincula con la T. La figura 22.3 le muestra una forma de representar tal vinculación. Este estilo en particular se conoce como *vinculación explícita* porque muestra de manera explícita la clase generada en una relación de dependencia con la clase parametrizada y le da a la clase generada su propio nombre.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 22.2**

*SerVivo como una clase parametrizada.*

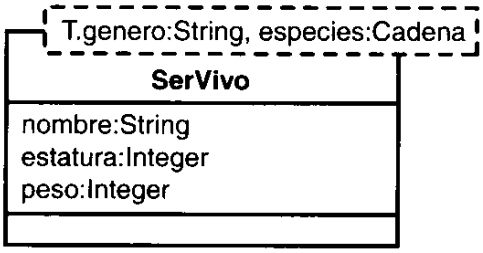
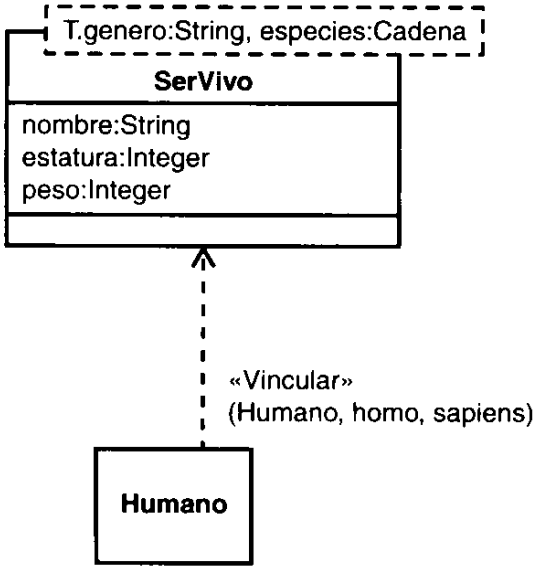



**FIGURA 22.3**

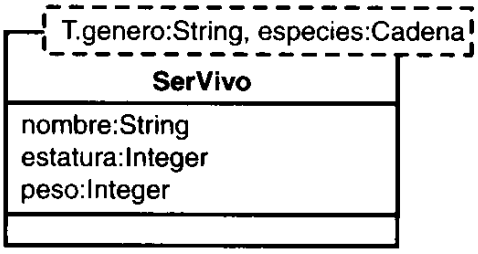
*Vinculación explícita de la clase parametrizada SerVivo.*




Hay otro tipo de vinculación que se conoce como *vinculación implícita*. En ella no se muestra la relación de dependencia, y las vinculaciones aparecen en una lista dentro de paréntesis angulares en el nombre de la clase generada. La figura 22.4 le muestra lo anterior.

**FIGURA 22.4**

*Vinculación implícita de la clase parametrizada SerVivo.*



En cualquier caso, podrá asignar valores a nombre, estatura y peso para generar objetos en la clase Humano.

www.elsolucionario.net

www.elsolucionario.net

# Patrones de diseño

Es posible expandir la idea de la parametrización. Cualquier clasificador UML puede ser parametrizado. De hecho, un grupo de clasificadores que colaboren entre sí pueden ser parametrizados, lo que nos evitará entrar en situaciones desconcertantes.

Después de varias décadas de amplio uso, mismo que se incrementa, la orientación a objetos ha otorgado varias soluciones firmes a los problemas recurrentes. Tales soluciones se conocen como ***patrones de diseño***, y han jugado un importante papel últimamente. Como los patrones de diseño han traspasado el mundo de la orientación a objetos, son fáciles de conceptualizar, diagramar y reutilizar. Al contar ahora con el UML, tenemos un lenguaje de modelado común para explicarlos y diseminarlos.

El primer libro que popularizó a los patrones de diseño fue "Design Patterns" (Addison-Wesley, 1995). Sus autores (Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides) han sido reconocidos como el "Grupo de los cuatro".

Un patrón de diseño es básicamente una solución (un diseño) que surge de la experimentación práctica con varios proyectos, y los equipos de desarrollo han encontrado que se puede aplicar en diversos contextos. Cada patrón de diseño describe a un conjunto de objetos y clases comunicadas. El conjunto se ajusta para resolver un problema de diseño en un contexto específico.

En su libro, el grupo de los cuatro catalogó y destacó a 23 patrones de diseño fundamentales. Dividieron esos patrones en tres categorías de acuerdo con cada uno de sus *propósitos*: (1) Patrones de *creación* que atañen al proceso de creación de objetos, (2) Patrones de *estructura* que se orientan a la composición de clases y objetos, y (3) Patrones de *comportamiento* que especifican la forma en que las clases u objetos interactúan y distribuyen la responsabilidad. Posteriormente dividieron sus patrones de diseño en términos si se aplican a las clases u objetos. A este criterio lo llamaron *ámbito*, y la mayoría de los ámbitos de los patrones se encuentra al nivel de los objetos.

Cada patrón de diseño tiene cuatro elementos: (1) un *nombre* que nos permite describir un problema de diseño en una palabra o frase, (2) un *problema* que define cuándo aplicar el patrón, (3) una *solución* que especifica los elementos que conforman al diseño y cómo colaboran, y (4) las *consecuencias* de aplicar el patrón.

Ahora veremos esas "situaciones desconcertantes" que indiqué con anterioridad: dentro de un modelo, podemos representar un patrón de diseño como una colaboración parametrizada en el UML. El patrón de diseño se expresa de una forma genérica, con nombres genéricos para los colaboradores. El asignar nombres específicos del dominio hace que el patrón se aplique a un modelo específico. La colaboración parametrizada le ayuda a visualizar los detalles específicos dentro del contexto del patrón.

www.elsolucionario.net

www.elsolucionario.net

# Cadena de responsabilidad

Examinemos un patrón de diseño y verá lo que quiero decir.

La Cadena de responsabilidad es un patrón de comportamiento que se aplica a cierta cantidad de dominios. Este patrón se encarga de la relación entre un conjunto de objetos y una petición. Aplicará este patrón cuando más de un objeto pueda manejar una petición. El primer objeto en la cadena obtiene la petición y la resolverá o se la enviará al siguiente objeto de la cadena, hasta que uno de ellos pueda manejarla. El objeto que originalmente hizo la petición no sabe cuál de los objetos la manejará. El objeto que al final maneje la petición se conoce como un *receptor implícito*.

Los restaurantes están establecidos de esta forma, al igual que los distribuidores de automóviles cuando financian la adquisición de automóviles. Por lo general, en un restaurante un cliente no envía una petición directamente a un chef ni sabe qué chef preparará su platillo. En vez de ello, el cliente le da una orden a un mesero, el mesero se la lleva al chef, quien podrá cumplir con ella o dársela a uno de sus asistentes (de cualquier forma, así ocurre con los supuestos restaurantes de LaHudra, Nar y Goniff). Con un distribuidor de automóviles, el agente de ventas pasa el pedido de préstamo a diversas instituciones financieras hasta que alguna decide otorgar el préstamo.

Las ligas de deportes profesionales implementan este patrón cuando un equipo pone transferible a un jugador. El equipo con la peor marca tendrá una oportunidad de solicitar al jugador. Si decide no hacerlo, el equipo con la siguiente peor marca tendrá la oportunidad, y así sucesivamente. El equipo que puso transferible al jugador no necesariamente sabe qué otro equipo se quedará con él (vea el Ejercicio 1).

Ahora que ha visto el patrón de diseño Cadena de responsabilidad en algunos contextos, ya está preparado para comprenderlo de manera abstracta. Los participantes en este patrón serán un Cliente, un Controlador abstracto y Controladores concretos que provengan del Controlador abstracto. El Cliente inicia una petición; si un Controlador (concreto) puede ocuparse de la petición, así lo hará; si no, pasará la petición al siguiente Controlador abstracto. La figura 22.5 le muestra la forma en que luce esta estructura.

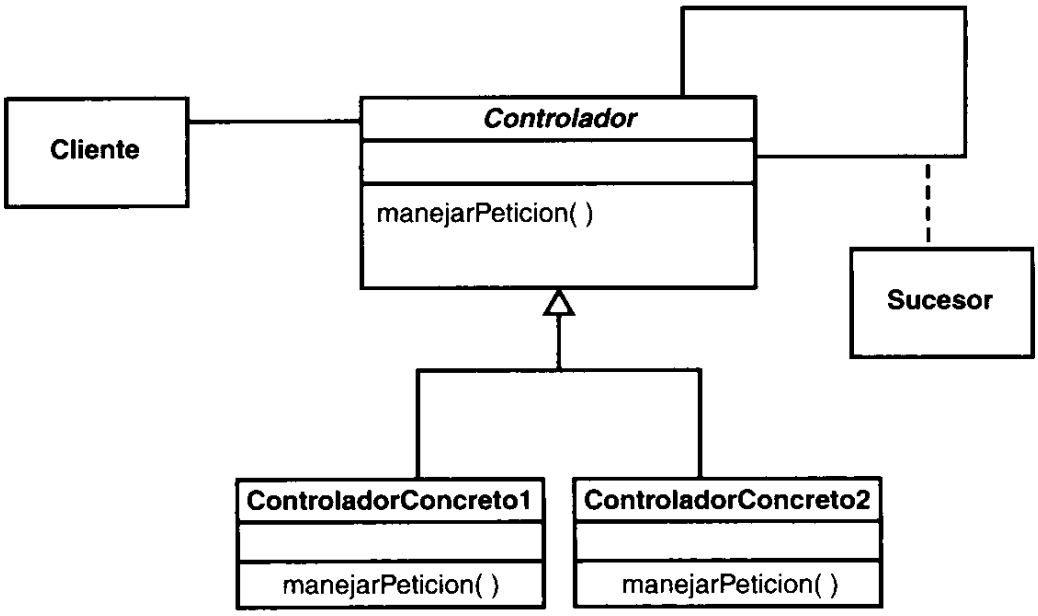
La idea de este patrón es liberar a un objeto de que tenga que saber qué otro objeto realizará su petición. Le da una flexibilidad adicional cuando asigne responsabilidades a los objetos. La desventaja es que el patrón no garantiza que algún objeto maneje la petición. Por ejemplo: ningún equipo de fútbol contrataría a un jugador que haya sido puesto como transferible.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 22.5**

*La estructura del patrón de diseño Cadena de responsabilidad.*




Observe la asociación reflexiva en la clase Controlador abstracto. El grupo de los cuatro intentó mostrar que usted tendrá la opción de que el Controlador implemente un vínculo sucesor (en algunos contextos, los objetos saben cómo encontrar a sus sucesores). He decidido representar tal implementación con una clase asociación como en la figura 22.5, para permitir que más adelante se puedan agregar atributos al sucesor.

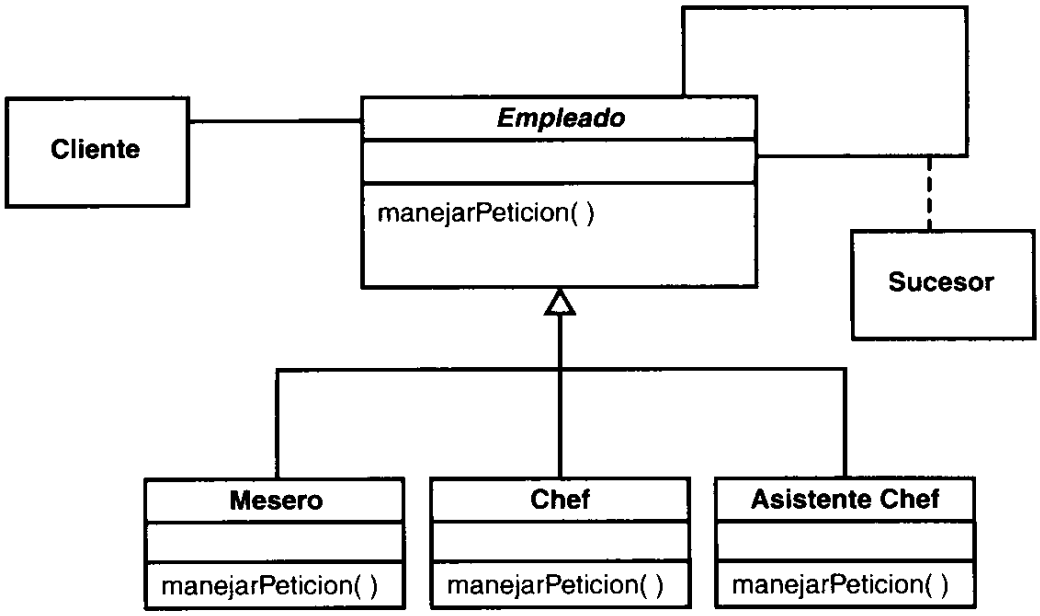
## Cadena de responsabilidad: dominio Restaurante

En el dominio Restaurante, el Controlador abstracto es la clase Empleado, y los Controladores concretos son el mesero, el chef y el asistente. El Cliente es, en sí, el propio cliente, quien podría iniciar una petición, como hacer una orden, y no saber quién la llevará a cabo.

Al sustituir los nombres específicos del dominio de la figura 22.5 nos deja la figura 22.6.

**FIGURA 22.6**

*El patrón de diseño Cadena de responsabilidad en el dominio del restaurante.*

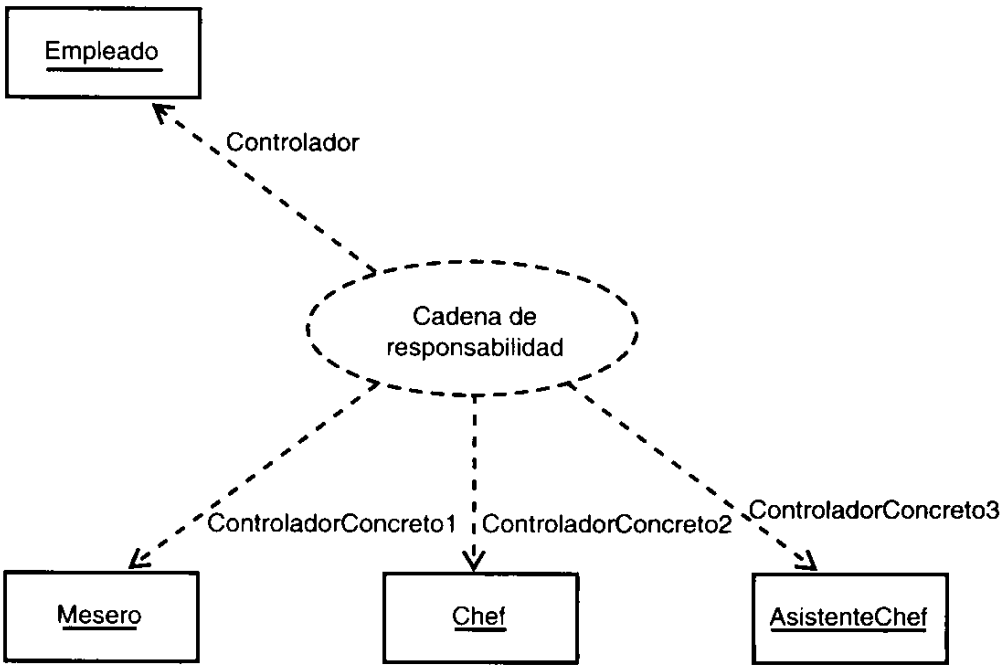



www.elsolucionario.net

La figura 22.6, aunque es un útil diagrama, no nos muestra la forma en que los nombres específicos del dominio encajan en el patrón. Para mostrar el contexto, utilizaremos una colaboración parametrizada, como en la figura 22.7.

**FIGURA 22.7**

*Una colaboración parametrizada para representar la Cadena de responsabilidad en un restaurante.*




En la figura 22.7, el óvalo punteado representa la colaboración que hay en el patrón de diseño, de allí el nombre dentro del óvalo. Los cuadros que lo rodean representan a los colaboradores. Las dependencias muestran que la colaboración depende de los colaboradores. La etiqueta en una dependencia indica el rol que juega el colaborador dependiente dentro del patrón. La colaboración se ha parametrizado con la adición de los nombres de clases específicas del dominio.

## Cadena de responsabilidad: Modelos de eventos de los exploradores Web

Cuando se desarrollan páginas Web interactivas, un diseñador debe tomar en cuenta el modelo de eventos del explorador que las abrirá. En Internet Explorer, puede escribir código de JavaScript o VBScript para reaccionar a un evento, como el clic en un botón. Este código, conocido como "controlador de evento", establece los cambios (en caso de haberlos) que habrá cuando se haga clic en un botón.

En un documento HTML, podrá dividir una página en áreas conocidas como DIV, y subdividir a un DIV en formularios. Puede colocar un botón dentro de un formulario. ¿Le suena a objeto compuesto? ¡Claro, porque eso es! Cada elemento es un componente del documento, y ciertos componentes son parte de otros. La Gamma lista a los objetos compuestos como uno de sus patrones de diseño, y se usa con frecuencia junto con el patrón de Cadena de responsabilidad. La relación entre componente y objeto compuesto implementa los vínculos de los sucesores. Cuando le mostré el diagrama de clases de la Cadena de responsabilidad, le mencioné a modo de aclaración que en ciertos contextos los objetos saben cómo localizar a sus propios sucesores. Éste es uno de esos contextos.

www.elsolucionario.net

www.elsolucionario.net

Cuando se coloca el botón en un formulario dentro de un DIV cuyo documento se abre en Internet Explorer, el evento de hacer clic en él iniciará con el botón en sí, pasará al formulario, luego al DIV y, finalmente, al documento que lo contiene. Cada uno de estos elementos puede tener su propio controlador del evento hacer clic en el botón, el cual reaccionará ante tal evento.

Si una secuencia de comandos que se encuentre en el documento establece de forma dinámica cuál de los controladores de eventos del elemento se ejecuta, la secuencia de comandos será una instancia del patrón de diseño de la Cadena de responsabilidad. La figura 22.8 le muestra el diagrama de clases, y la 22.9 le muestra la colaboración parametrizada para este patrón de diseño aplicada al modelo de eventos del Internet Explorer.

**FIGURA 22.8**

*Diagrama de clases de la Cadena de responsabilidad en una página Web que se abre en Internet Explorer.*

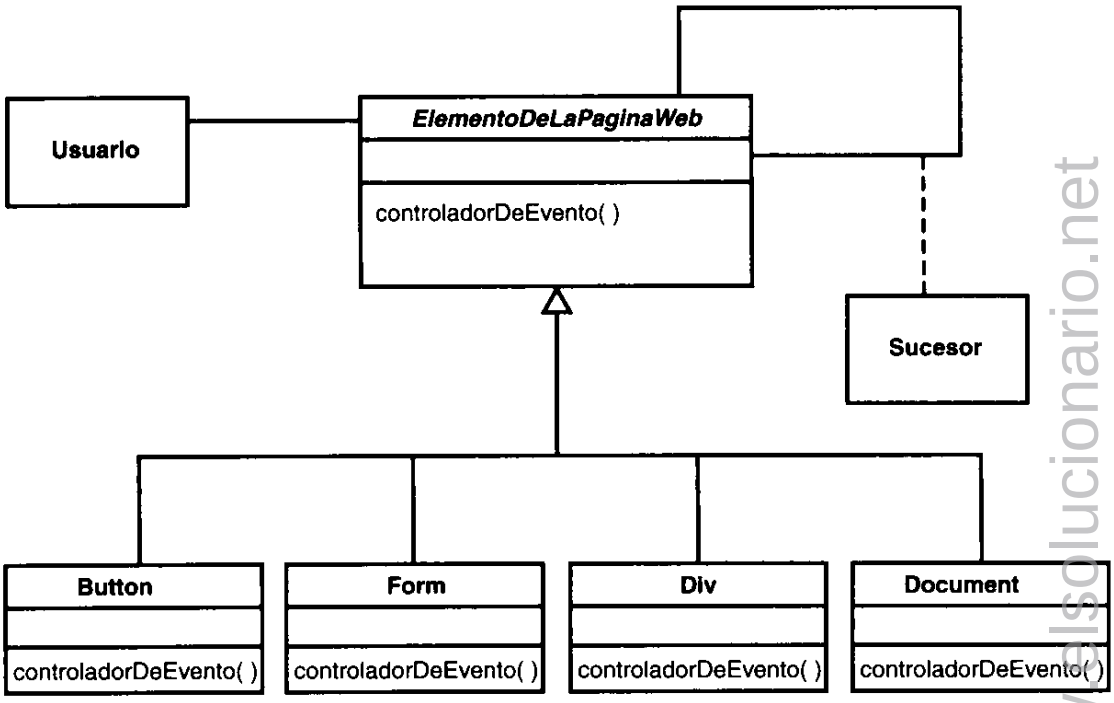
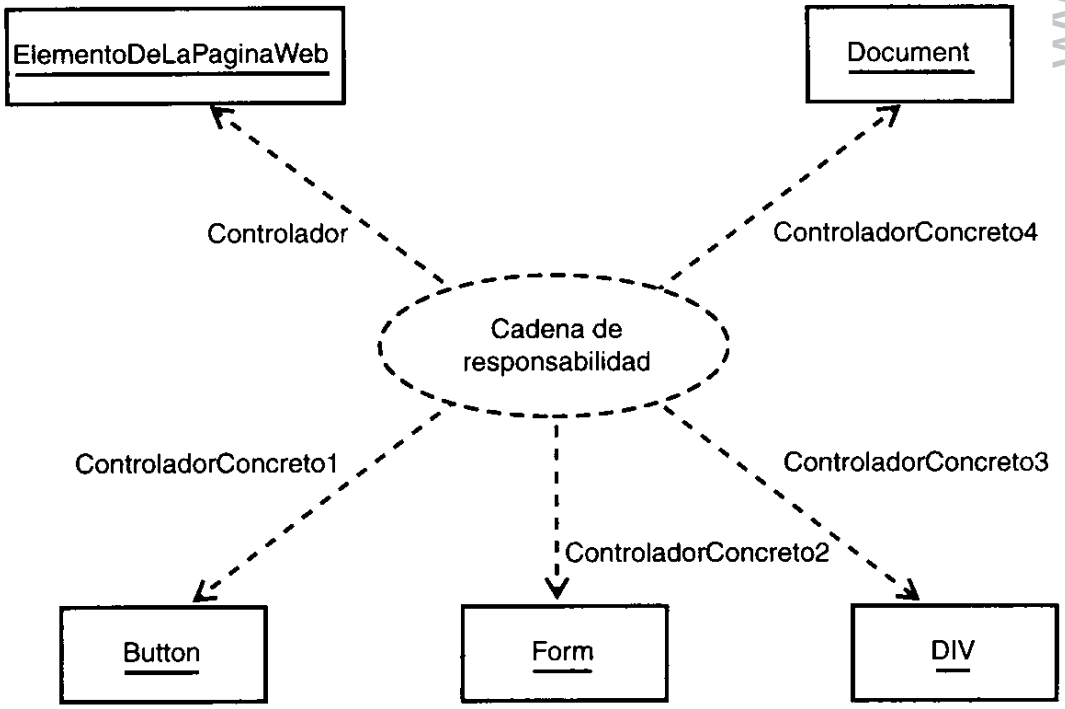



**FIGURA 22.9**

*La Colaboración parametrizada para la Cadena de responsabilidad en una página Web que se abre en Internet Explorer.*




www.elsolucionario.net

Netscape Navigator también cuenta con un modelo de eventos. Su modelo es, exactamente, lo opuesto al de Internet Explorer. En Navigator, el elemento de mayor nivel (el documento) primero obtiene el evento y lo pasa hasta que lo lleva al que lo originó. ¿Cómo cambiaría al diagrama de clases de la figura 22.8 para modelar al modelo de eventos de Navigator? (Vea el Ejercicio 2.)

Internet Explorer llama a su modelo de eventos *burbujeo de eventos*. Navigator lo llama *captura de eventos*.

# Nuestros propios patrones de diseño

Aunque el Grupo de los cuatro saltaron a la fama por su catálogo de patrones de diseño, no establecieron que sus patrones fueran los únicos posibles. Al contrario, intentaron alentar el descubrimiento y uso general y propio de los patrones.

Para darle una idea de la forma en que surgen tales patrones, recuerde lo que hicimos en la hora 11, "Diagramas de actividades". Durante esa hora, vio un ejemplo de cómo tratar con los números de Fibonacci y un ejercicio referente a los números triangulares.

¿Cuáles fueron las características en común? Cada uno empezó con un valor o conjunto de valores iniciales, siguió con una regla para acumular números y finalizó con el enésimo número de la serie.

Llamemos a este patrón "Calculadora de series". Aunque podríamos implementarlo en un objeto, hagámoslo en un conjunto de objetos colaboradores para ilustrar algunos conceptos de los patrones de diseño.

La Calculadora de series cuenta con tres participantes: ValorInicial (que puede contener uno o más valores), ReglaDeAcumulacion y ValorFinal. La figura 22.10 muestra el diagrama de clases de este patrón. El valor inicial está en un atributo llamado *primero*. Si es necesario un segundo valor inicial, como en el caso de los números de Fibonacci, se especifica en un atributo llamado *segundo*. En ocasiones, como en el caso de los factoriales, el patrón necesitará un valor para el término *cero*. El algoritmo para la ReglaDeAcumulacion se implementa en la operación acumular(). La cantidad de términos por calcular está en el *enésimo* atributo en ReglaDeAcumulacion.

En la colaboración, ReglaDeAcumulacion obtiene el (los) valor(es) inicial(es) de ValorInicial, aplica la regla la cantidad de veces indicada y envía el resultado a ValorFinal para que sea mostrado. La figura 22.11 muestra la interacción.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 22.10**

*La estructura de clases para nuestro patrón de diseño Calculadora de series.*

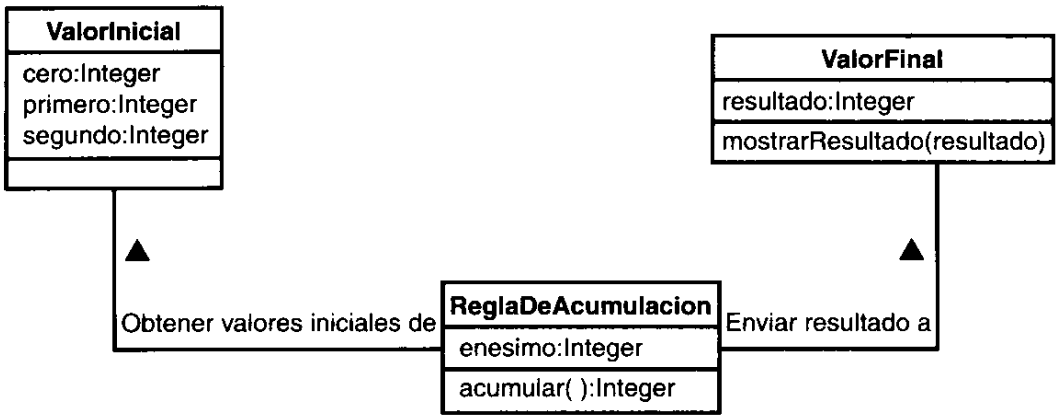
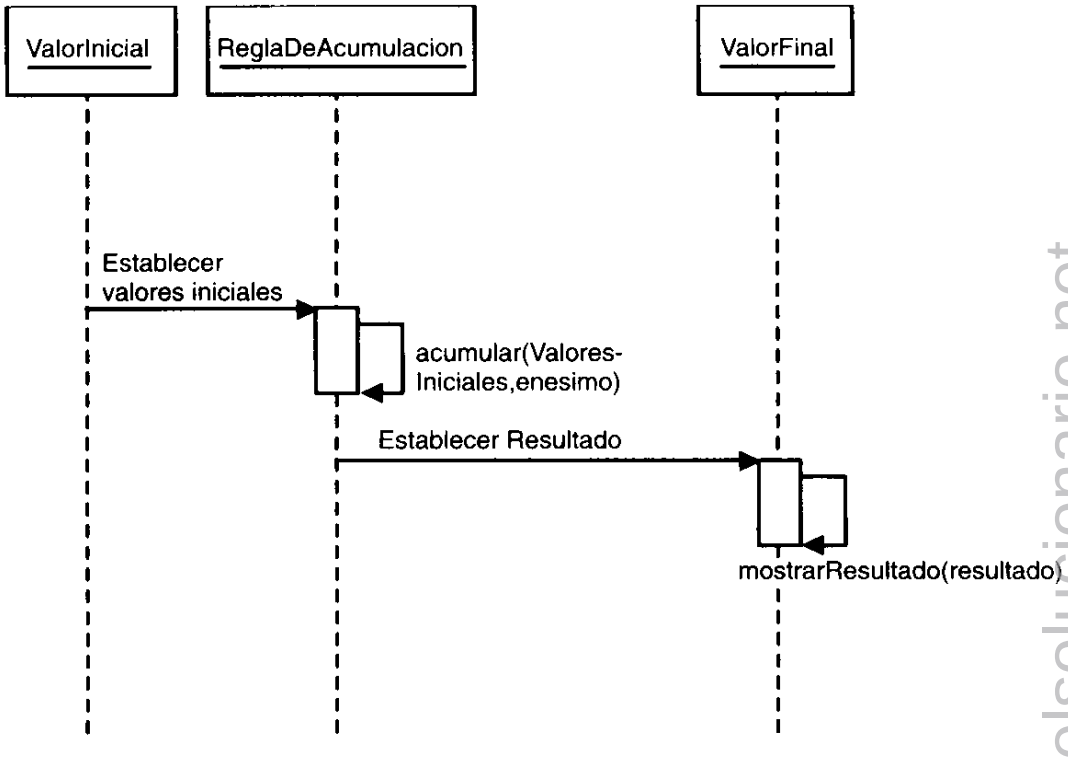



**FIGURA 22.11**

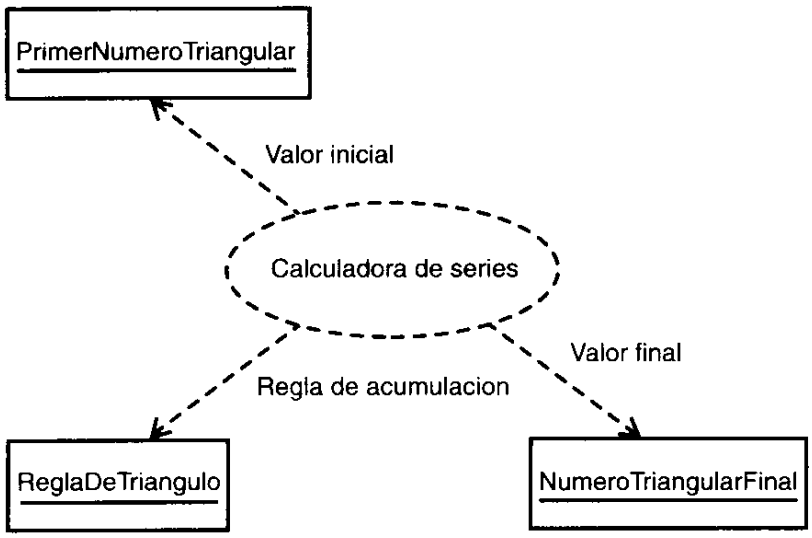
*La interacción dentro del patrón de diseño Calculadora de series.*




Para aplicar este patrón de diseño a la serie de números triangulares, adoptaremos algunos nombres de la numeración triangular para las clases y mostraremos la colaboración parametrizada de la figura 22.12.

**FIGURA 22.12**

*La colaboración parametrizada para la Calculadora de Números triangulares.*

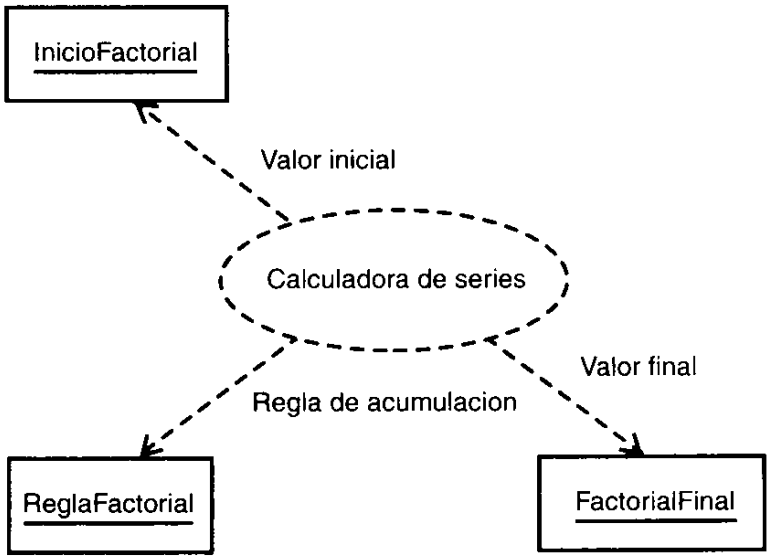



www.elsolucionario.net

www.elsolucionario.net

Como buena medida, mostraremos la colaboración parametrizada para una calculadora de factorial en la figura 22.13.

**FIGURA 22.13**
*La colaboración parametrizada para una calculadora de Factorial.*




# Ventajas de los patrones de diseño

Los patrones de diseño son útiles en diversas maneras. Para empezar, promueven la reutilización. Si ha expresado un diseño bien fundamentado como patrón, lo habrá hecho sencillo para usted y otros que con él trabajen nuevamente. A su vez, le dan una forma clara y concisa de hablar y pensar respecto a un grupo de clases u objetos que funcionen en conjunto para resolver un problema. Esto aumenta la posibilidad de que utilice al patrón como un componente de un diseño. Finalmente, si utiliza patrones en su diseño, posiblemente se le facilitará documentar el sistema que haya generado.

# Resumen

Una clase parametrizada tiene parámetros desvinculados. El vincular a tales parámetros dará como resultado la creación de una clase. Cualquier clasificador UML podrá estar parametrizado. Una colaboración parametrizada sirve como la representación de un patrón de diseño: una solución que es útil en diversos dominios.

Un patrón de diseño, la "Cadena de responsabilidad", se ocupa de que los objetos pasen una petición entre ellos hasta que uno pueda manejarla. Este patrón proviene del libro más conocido en patrones de diseño, escrito por un grupo de autores conocidos como "El grupo de los cuatro".

Nuestros propios patrones de diseño surgen del trabajo que hicimos en la hora 11 en los diagramas de actividades. Podemos crear un patrón de diseño para una calculadora que determine el enésimo valor de una serie aritmética. Los participantes en este patrón son ValorInicial, ReglaDeAcumulacion y ValorFinal.

www.elsolucionario.net

www.elsolucionario.net

Los patrones de diseño ofrecen varias ventajas. Permiten que los diseñadores vuelvan a utilizar con facilidad las soluciones ya probadas, incorporar componentes sólidos en los diseños y documentar claramente los sistemas que generen.

# Preguntas y respuestas

**P ¿Qué tan difícil es "descubrir" los patrones de diseño?**

**R** No es cuestión de dificultad, sino de experiencia. Conforme avance en su carrera de analista y diseñador, verá que hay ciertas cosas que se repiten una y otra vez y después de un tiempo tratará de aprovechar esas circunstancias regulares. Los estudios muestran que los expertos en un dominio en particular piensan en términos de patrones y tratan de aplicarlos siempre que pueden.

**P ¿Los patrones sólo son útiles en el diseño?**

**R** No. Los patrones pueden surgir en cualquier momento del proceso de desarrollo o en cualquier campo de acción. El Grupo de los cuatro se inspiró en el trabajo de un arquitecto que pensó en recurrir a patrones para el diseño de los edificios.

# Taller

Las preguntas y ejercicios en este taller le llevarán a pensar respecto a algunas características avanzadas del UML. Vaya al Apéndice A, "Respuestas a los cuestionarios", para encontrar las respuestas.

## Cuestionario

1. ¿Cómo representaría a una clase parametrizada?
2. ¿Qué es "vincular" y cuáles son los dos tipos de vinculación?
3. ¿Qué es un "patrón de diseño"?
4. ¿Qué es el patrón de diseño "Cadena de responsabilidad"?

## Ejercicios

1. Aplique el diagrama de clases del patrón diseño Cadena de responsabilidad al proceso de transferencia que se realiza en las ligas deportivas profesionales.
2. Modifique el diagrama de clases de la figura 22.8 de modo que deje entrever el modelo de eventos de Netscape Navigator. Como ya lo indiqué anteriormente, un evento en Navigator se inicia al nivel del documento y pasa por los distintos elementos hasta que llega al que lo originó. El elemento originador podría localizarse a varios niveles por debajo del documento HTML.

www.elsolucionario.net

www.elsolucionario.net

# PARTE III
# Visión del futuro

## Hora

23 Modelado de sistemas incrustados

24 El futuro del UML

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# HORA 23

# Modelado de sistemas incrustados

Esta vez verá lo referente a sistemas de cómputo que no se basan en sistemas de escritorio, laptops o en palmtops. En vez de ello, estarán incrustados dentro de aparatos como aviones, trenes y automóviles.

En esta hora se tratarán los siguientes temas:

- ¿Qué son los sistemas incrustados?
- Conceptos de los sistemas incrustados
- Modelado de un sistema incrustado en el UML

LaHudra y sus intrépidos socios, Nar y Goniff, han tenido buenas ganancias de su División de Restaurantes LNG. El servicio es tan bueno y las comidas tan deliciosas que la gente llega por miles para probarlas dentro de una atmósfera de eficiencia y amistad.

www.elsolucionario.net

www.elsolucionario.net

Hay dos problemas que deterioran su otrora buena fortuna. Conforme leen los informes mensuales, la terrible tendencia ha continuado. "Miren esto", dijo Nar dándoles los escritos a Goniff y LaHudra. "Estamos haciendo mucho dinero, pero deberíamos hacer más. Parece que los meseros están rompiendo más losa de la aceptable."

"Sí, ya lo había advertido", dijo Goniff. "Cada vez que rompen un plato lleno de comida, el cocinero tiene que preparar otra vez el plato. Y tenemos que pagar el nuevo plato."

"¿Realmente importa si algunos de nuestros meseros tienen las manos torpes?", preguntó LaHudra.

"Claro que sí", respondió Goniff. "Un par de platos aquí, otros allá, pronto las pérdidas serán considerables. Pero hay algo de los meseros que me está molestando más."

"¿Y qué es?", preguntó Nar.

"Los reportes indican que se están enfermando demasiado. Es bueno eso de que tengamos toda esta tecnología en los restaurantes. Nos ayuda cuando tenemos que trabajar con limitaciones en el personal: por lo general los meseros pueden hacerse cargo de mayores secciones de lo que podrían normalmente."

"Veamos qué ocurre", dijo LaHudra.

# La madre de la invención

Los tres socios entrevistaron a varios de los meseros que se habían enfermado con frecuencia en los dos últimos meses, e hicieron un asombroso descubrimiento: los platos rotos y los días de enfermedad estaban relacionados. Los meseros habían estado manejando y empuñando sus palmtops tanto que sus muñecas habían comenzado a debilitarse. Tal como una saliente puede hundir embarcaciones, una muñeca débil puede dejar caer los platos. Lo que es más, habían tenido tanto dolor en sus muñecas que no podían ir a trabajar.

"¿No podríamos ayudar a estas personas de alguna forma?", preguntó desconsolado Nar.

"¿Y en el proceso ayudarnos a nosotros mismos?", contra argumentó el oportunista Goniff.

"Tal vez haya alguna forma de que les ayudemos a mejorar sus fuerzas y sus muñecas", dijo LaHudra.

"Bien, ¿qué haremos?", preguntó Goniff, "¿comprar a cada uno un ejercitador de muñecas?"

"Podría ser peor", dijo LaHudra, "pues no sé qué tan efectivos sean tales ejercitadores. Podría ser eterno el proceso de curación de nuestro personal".

www.elsolucionario.net

www.elsolucionario.net

"Con todo, la idea es buena", dijo Nar, "tal vez sólo necesitemos un ejercitador de muñecas mejor que el que se pudiera adquirir en algún establecimiento".

"¿En verdad? ¿Cómo podríamos hacer un mejor ejercitador de muñecas?", preguntó LaHudra.

No tuvo que esperar mucho la respuesta. Nar estaba en uno de sus papeles de querer patentar.

"Recuerdo que mucha gente cree que la mejor y más eficiente forma de ejercitarse es aquella que genera el mayor reto, cuando sus músculos están trabajando a su máxima potencia. Si podemos crear un ejercitador de muñecas que aumente su resistencia conforme los músculos del antebrazo trabajen más, apuesto que mejoraremos la fuerza de las muñecas de nuestros meseros en la mitad del tiempo que ocuparían con un ejercitador de muñecas común."

"Y, ¿cómo haríamos eso?" se preguntaba el siempre pragmático LaHudra.

"Tal como revolucionamos al negocio de los restaurantes", dijo Nar, "con tecnología".

"A ver, espérame", dijo LaHudra, "lo que hicimos en los restaurantes requirió el uso de computadoras. ¿Realmente quieres decirme que agregaremos una computadora a un ejercitador de muñecas?"

"¿Y por qué no?" dijo Nar.

"¡Cierto! ¿Por qué no?", replicó Goniff. "Ya te entendí, Nar. Y cuando terminemos de generar este aparato, podríamos comercializarlo. Y ya tengo el nombre perfecto: ¿Qué tal 'LNG TecnoApretón'?"

"Creo que me está gustando", dijo, con cautela, LaHudra.

"Ya me gusta", indicó Nar, con entusiasmo. "¿Dónde está ese equipo de desarrollo?"

www.elsolucionario.net

# Creación de TecnoApretón

El equipo de desarrollo de RICO se ha vuelto a reunir. Su nueva misión será la de generar TecnoApretón, una pulsera "inteligente" para la muñeca y el antebrazo que dé una resistencia variable durante los movimientos repetitivos de un ejercicio: cuanto más trabajen los músculos, mayor será la presión de TecnoApretón.

Durante la realización de la idea, el equipo hizo algunas investigaciones para ver cómo medir qué tanto trabaja un músculo. Aprendieron lo referente a señales eléctricas de las fibras activas de los músculos. Tales señales, llamadas EMG (siglas de señales Electromiográficas), son el fundamento de fascinantes dispositivos que permiten a los discapacitados manejar equipo electrónico.

www.elsolucionario.net

Esto no será un tratado de ciencia-ficción. En los primeros años de la década de los noventa, el neurocientífico David Warner del Centro Médico de la Universidad Loma Linda colocó electrodos en el rostro de un muchacho y lo conectó a una computadora. El muchacho, completamente paralizado del cuello para abajo en un accidente automovilístico, pudo mover objetos en la pantalla de la computadora al tensar algunos de sus músculos faciales.

Para aprender más respecto a esta emocionante área, lea el artículo de Hugh S. Lusted y Benjamin Knapp, "Controlling Computers with Neural Signals", en la revista de octubre de 1996 de *Scientific American*.

El equipo concluyó que podrían capturar estos EMGs mediante pequeños y económicos "electrodos de superficie" colocados en el antebrazo, pasar los EMGs capturados por una computadora y, luego, utilizarlos como base para que la computadora ajuste la resistencia en el ejercitador de muñecas. Esto trae consigo la captura y análisis de datos en tiempo real, dado que los ajustes tienen que hacerse tan pronto como el músculo se contrae.

Una posibilidad del diseño sería colocar el electrodo de superficie en el antebrazo, conectarlo a una computadora de escritorio para que analice los EMGs y haga los ajustes necesarios al ejercitador de muñecas. La ventaja sería que posibilitaría la presentación de diversos datos en la pantalla, imprimir informes del progreso, y analizar las tendencias. No obstante, la desventaja sería que la persona quedaría atada a la computadora.

Otra posibilidad es la de incrustar un chip de computadora directamente en el ejercitador de muñecas para que la persona pueda moverse a donde desee mientras utilice TecnoApretón. La figura 23.1 muestra cómo luciría este diseño. En cada repetición del ejercicio, la persona aprieta la barra de constricción y la mueve hacia la barra base.

La ventaja del diseño incrustado sería que el usuario podría utilizar un dispositivo como éste en casi cualquier lugar (si la computadora tiene suficiente energía en las baterías). La desventaja, claro, es la pérdida de la información potencial que pudiera almacenar y mostrar una computadora de escritorio.

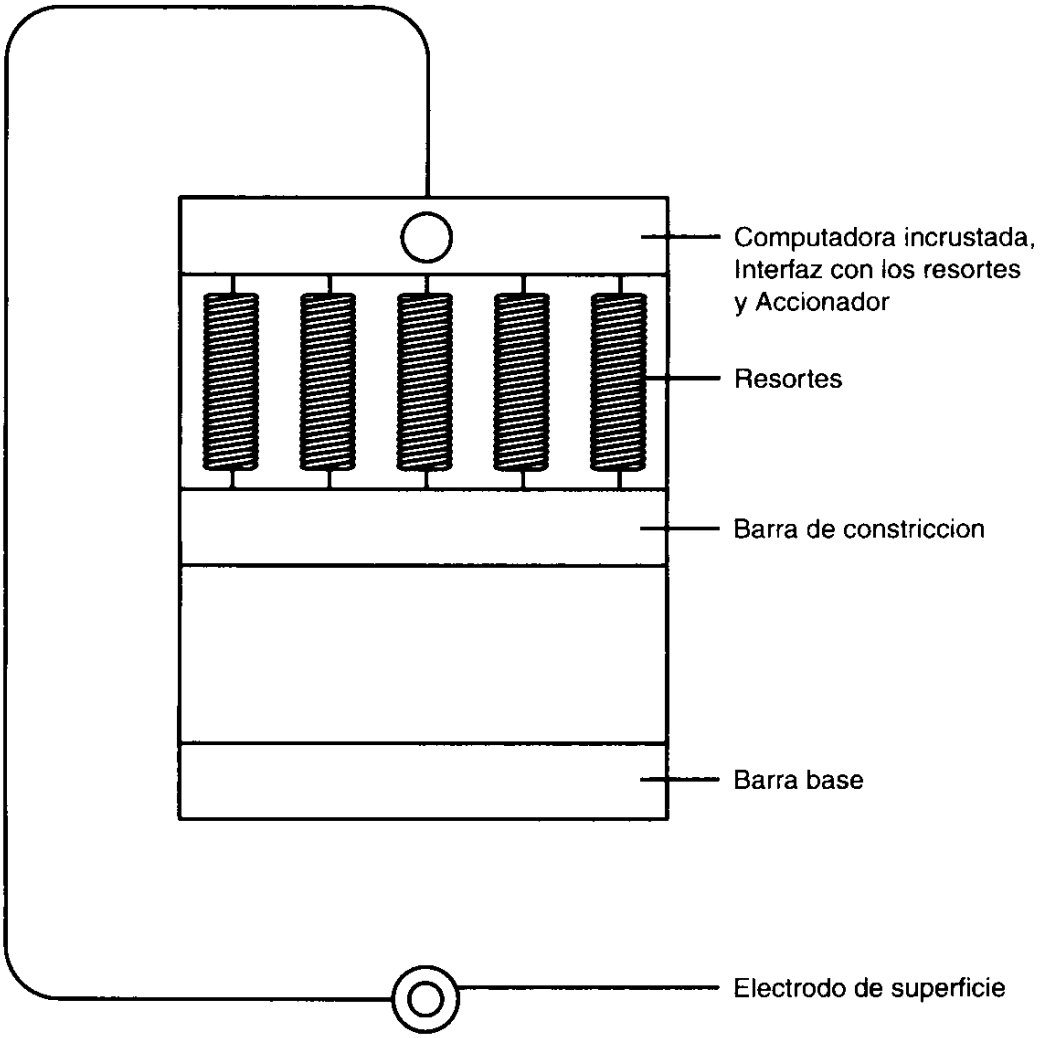
Las sesiones JAD dejaron entrever que todos serían más felices con el segundo diseño, y ello nos lleva hacia el mundo maravilloso de los sistemas incrustados.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 23.1**

*La versión en sistema incrustado de TecnoApretón.*




# ¿Qué es un sistema incrustado?

A estas alturas, ya sabe que las computadoras están por doquier. Lo que tal vez no sepa es qué tanto territorio abarca ese "doquier". Las computadoras que ve a su alrededor son sólo la punta del témpano. Muchas de ellas se encuentran ocultas en lugares de difícil acceso: dentro de los electrodomésticos, automóviles, aeroplanos, maquinaria de fábrica, dispositivos biomédicos y otros. Hay procesadores medianamente poderosos dentro de las impresoras.

Todas esas computadoras que no se ven a simple vista son ejemplos de sistemas incrustados. Siempre que tenga un dispositivo "inteligente", tendrá un sistema incrustado.

Los sistemas incrustados no tienen teclados y monitores que interaccionen con nosotros, en vez de ello, cada uno es un chip que se encuentra dentro de un dispositivo (como un aparato electrodoméstico), y ese dispositivo no se parece para nada a una computadora. El sistema incrustado decidirá qué debe hacer el dispositivo.

Si utiliza un sistema de este tipo, no tendrá la sensación de trabajar con una computadora. En vez de ello, estará interactuando con un dispositivo. Le apuesto que nunca sabrá que hay un chip de computadora dentro. Cuando tuesta una rebanada de pan, no le preocupa que un chip de computadora esté distribuyendo el calor —sólo quiere su pan tostado.

www.elsolucionario.net

www.elsolucionario.net

Cuando termina de trabajar con su computadora, la apaga. Un sistema incrustado por lo general no se puede dar esos lujos. Una vez que se le coloca, un sistema tiene que trabajar por días o años (como en un marcapasos).

Si un procesador de textos o una hoja de cálculo tienen un error y su sistema se colapsa, simplemente vuelve a arrancarlo. Si el software en un sistema incrustado falla, los resultados pueden ser desastrosos.

Por lo tanto, un sistema incrustado no realiza el cómputo de la forma habitual. Se le coloca para ayudar a otro tipo de dispositivo a hacer su trabajo. El otro dispositivo es el que es manejado por el usuario y se integra al entorno.

Como podrá imaginar, programar un sistema incrustado no es para novatos. Requiere mucho conocimiento del dispositivo donde se encontrará el sistema: qué tipo de señales enviará, qué tipo de parámetros de cronometraje tendrá y otras cosas.

# Conceptos de los sistemas incrustados

Veamos de cerca los sistemas incrustados y lo que comúnmente tienen que hacer. En las siguientes subsecciones examinaremos algunos de los conceptos más importantes de un sistema incrustado.

## Tiempo

Si revisa el tema hasta donde vamos, verá que el *tiempo* se destaca en el mundo de los sistemas incrustados. De hecho, el tiempo es la base para clasificar a los sistemas incrustados como *tolerantes* o *estrictos.*

Un sistema tolerante hace su trabajo tan pronto como sea posible sin tener que cumplir con plazos específicos. Un sistema estricto también tiene que hacer su trabajo tan pronto como sea posible, pero deberá finalizar sus tareas de acuerdo con plazos rigurosos.

## Subprocesos

En el mundo de los sistemas incrustados, un subproceso (que también se conoce como *tarea*) es un simple programa. Es una sección de una aplicación, y realiza cierto trabajo significativo dentro de ella. Intenta obtener toda la atención de la CPU. La *multitarea* es el proceso de programar el tiempo de la CPU para que trabaje con varios subprocesos y atienda a uno y otro.

Cada subproceso cuenta con un número que establece su prioridad dentro del programa de aplicación, y que se relaciona —por lo general— con uno de seis estados:

- *inactivo:* está en la memoria y no se ha hecho disponible al sistema operativo
- *listo*: puede ejecutarse, pero el subproceso que está en ejecución tiene una prioridad mayor

www.elsolucionario.net

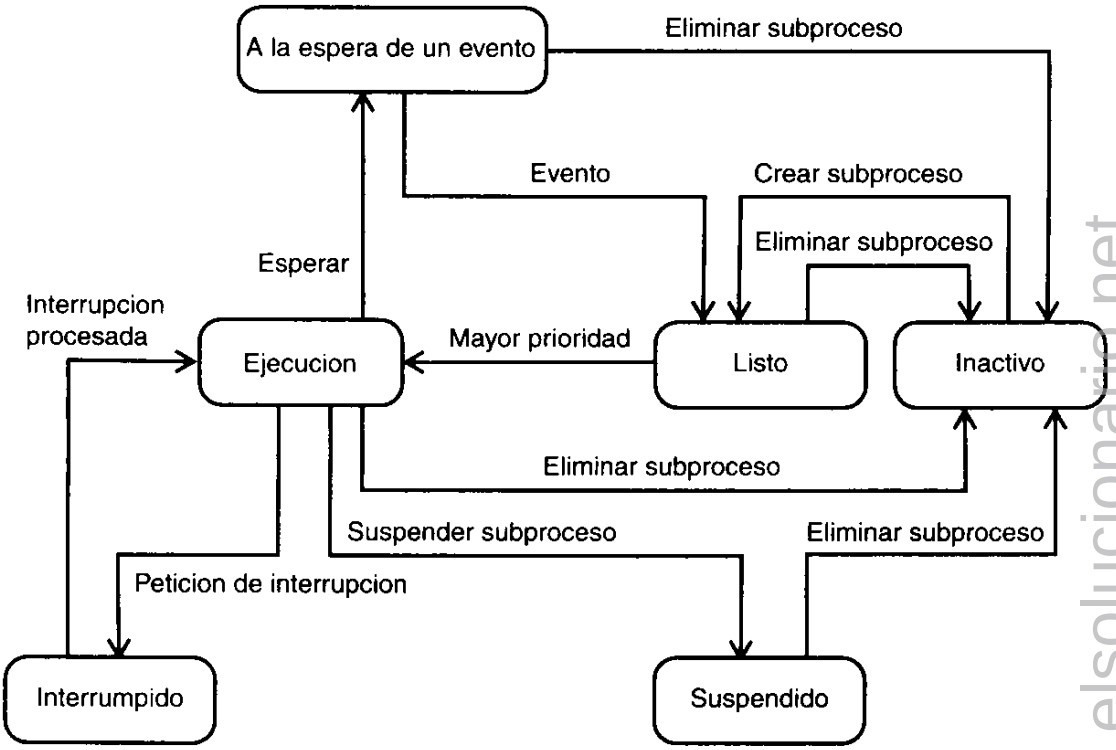
www.elsolucionario.net

- *suspendido*: se ha interrumpido a sí mismo por cierto tiempo
- *a la espera de un evento*: algo debe ocurrir para que se ejecute
- *ejecución*: cuando está siendo atendido por la CPU
- *interrumpido*: poque la CPU se está ocupando de una *interrupción*

La figura 23.2 le muestra un diagrama de estados que representa a estos estados y las transiciones entre ellos. Observe la ausencia de un símbolo de inicio y otro de fin. Esto le indica que el subproceso fluye de un estado a otro en un ciclo infinito.

**FIGURA 23.2**

*Estados de un subproceso en una aplicación de un sistema incrustado.*




Por cierto, ¿qué es una "interrupción"? Continúe leyendo.

## Interrupciones

Una *interrupción* es un elemento pequeño de mucha importancia en un sistema incrustado. Es un mecanismo basado en hardware que le indica a la CPU que ha acontecido un evento asincrónico. Un evento es asincrónico si sucede de forma impredecible (esto es, fuera de sincronía). Por ejemplo: en el TecnoApretón, las señales EMG llegan de forma asincrónica.

Cuando la CPU reconoce una interrupción, guarda lo que estaba haciendo e invoca a un ISR (Rutina para el servicio de interrupciones) que procesa al evento. Cuando el ISR finaliza su trabajo, la CPU continúa con lo que hacía cuando sucedió la interrupción.

www.elsolucionario.net

www.elsolucionario.net

Después de procesar una interrupción, el lugar a donde regresará la CPU se determina por el tipo de sistema operativo que en ella se ejecuta, como lo verá posteriormente.

Las interrupciones son importantes dado que permiten que una CPU se libere de lo que está haciendo y procese los eventos conforme suceden. Esto es tremendamente significativo para un sistema en tiempo real que tenga que responder a eventos del entorno de manera oportuna.

Ya que la puntualidad es tan importante, los sistemas incrustados tienen que ocuparse del curso del tiempo en una interrupción y su procesamiento, aunque tal lapso pudiera parecer infinitesimal. La CPU tiene que tomar algo de tiempo desde que se le notifica de la interrupción hasta que guarda lo que estaba haciendo (esto es, su *contexto*). A ello se le conoce como *latencia de la interrupción*. La *respuesta de la interrupción* es el tiempo entre la llegada de la petición de interrupción y cuando la CPU inicia el ISR. Al finalizar el ISR, *la recuperación de la interrupción* es el tiempo que le lleva a la CPU regresar a donde estaba —a su contexto— cuando ocurrió la interrupción.

Hay un tipo de interrupción especial: los *ciclos del reloj*. Como un "latido de corazón" del sistema, el ciclo del reloj sucede en intervalos regulares específicos de una aplicación (por lo general entre 10 y 200 microsegundos). Los ciclos del reloj determinan las restricciones de tiempo de un sistema incrustado. Por ejemplo: un subproceso en estado suspendido permanecerá así durante una cantidad específica de ciclos.

## Sistema operativo

Un sistema operativo en tiempo real (RTOS) funciona como un policía de tránsito entre los subprocesos y las interrupciones; media la comunicación entre subprocesos y entre una interrupción y un subproceso. El núcleo o *kernel* es la parte del *RTOS* que administra el tiempo que utiliza la CPU en subprocesos individuales. El *programador* del núcleo determina cuál subproceso se ejecuta después. Como lo mencioné, cada subproceso tiene una prioridad asignada.

Los núcleos pueden ser *preferenciales* o *cooperativos* (no preferenciales), de acuerdo con la forma en que se encarguen de las interrupciones. En un núcleo cooperativo cuando se termina la ejecución de un ISR, la CPU regresa al subproceso en que se encontraba cuando llegó la petición de interrupción. Un núcleo cooperativo se dice que se involucra en la multitarea cooperativa. La figura 23.2 se aplica al núcleo cooperativo.

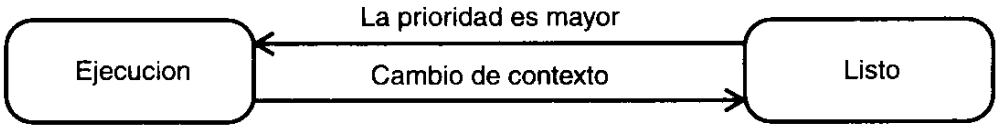
Por otro lado, en un núcleo preferencial, cuando finaliza un ISR, la CPU verifica la prioridad de los subprocesos que se encuentren en el estado de Listo. Si uno de los subprocesos tiene una mayor prioridad que la tarea interrumpida, la CPU ejecutará dicho subproceso en lugar de aquél en el que se encontraba cuando se recibió la petición de interrupción. Por ello, la tarea de mayor prioridad tiene preferencia sobre la tarea interrumpida. La figura 23.3 muestra la modificación a dos de los estados en la figura 23.2, para modelar al núcleo preferencial.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 23.3**

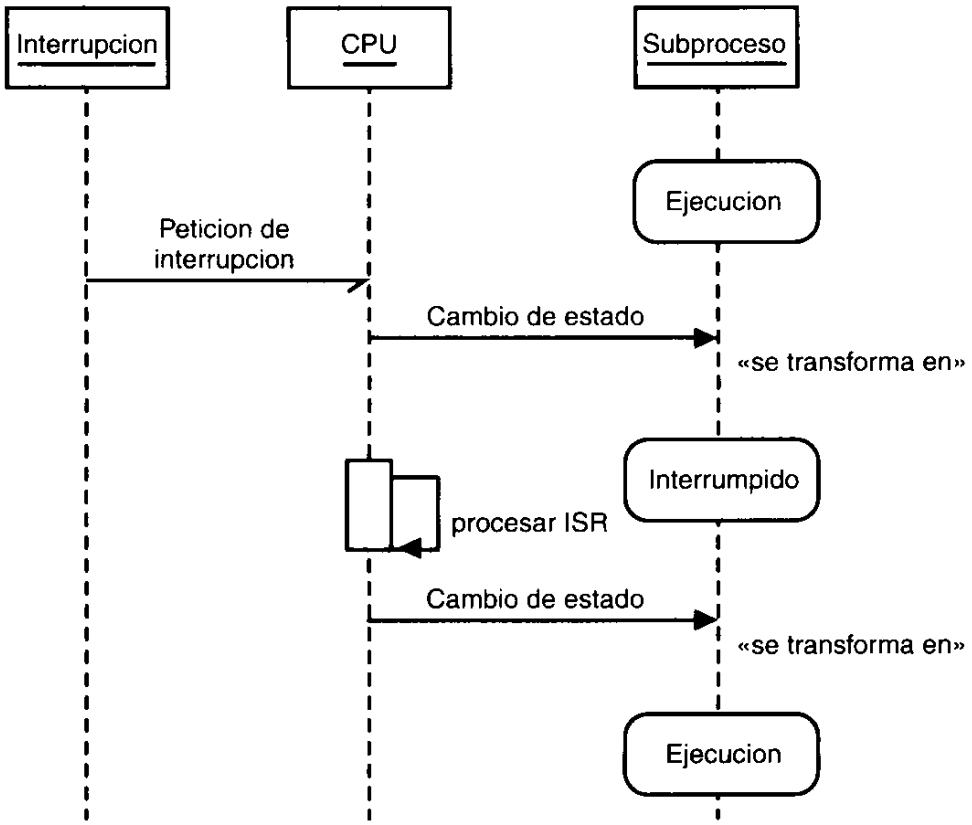
*Modificación de transiciones entre dos estados en la figura 23.2 para reflejar lo que ocurre en un núcleo preferencial.*




La Figura 23.4 modela el núcleo cooperativo como un diagrama de secuencias, y la figura 23.5 hace lo propio para el núcleo preferencial.

**FIGURA 23.4**

*Diagrama de secuencias para el núcleo cooperativo.*




www.elsolucionario.net

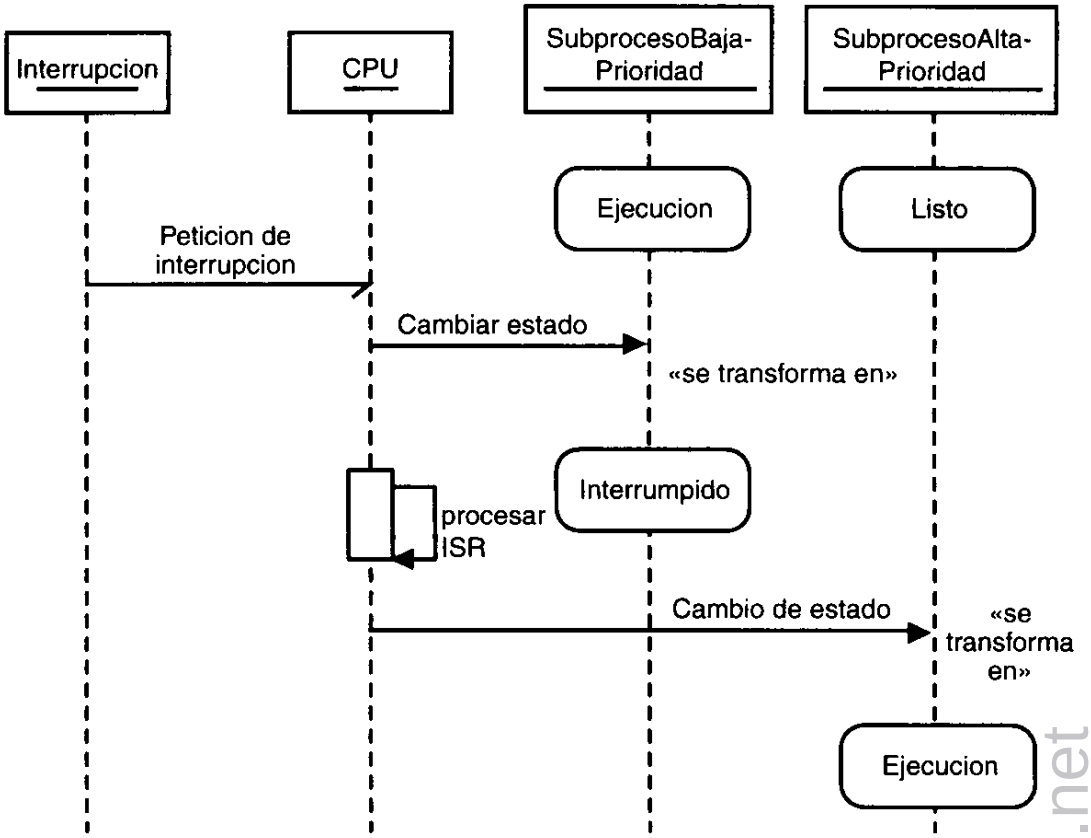
Microsoft empezó a reducir la brecha hacia el dominio de los sistemas incrustados. Está comercializando, de Windows CE, el sistema operativo para dispositivos de mano, al igual que para sistemas incrustados. Windows CE está generado de forma modular. Para ajustarse a una CPU en particular y a una aplicación incrustada, sólo requiere utilizar los módulos que necesite para que funcione.

La ventaja de utilizar Windows CE es que podrá utilizar el popular conjunto de herramientas de Microsoft junto con Visual C++ para generar un sistema incrustado.

www.elsolucionario.net

**FIGURA 23.5**

*Diagrama de secuencias para el núcleo preferencial.*




Aunque hemos visto bastantes fundamentos, tenga en cuenta que sólo hemos visto cuestiones superficiales de los sistemas incrustados.

# Modelado de TecnoApretón

De regreso a la tarea (¿subproceso?) que dejamos suspendido, empezaremos a crear un modelo para TecnoApretón. Aunque no es el caso de que todos los sistemas incrustados estén orientados a objetos, aun podemos utilizar tal orientación para modelar al sistema y su comunicación con el mundo exterior.

A partir de nuestro tema de los sistemas incrustados, es claro que tenemos que tomar en cuenta el cronometraje, los eventos, los cambios de estado y las secuencias.

## Clases

Como en el caso de cualquier otro tipo de sistema, empezaremos con las clases. Para comprender la estructura de clases, empezaremos con una descripción resumida de TecnoApretón y la forma en que funciona. Este resumen pudo haber sido el resultado de un análisis del dominio.

He aquí la descripción. TecnoApretón consta de un electrodo de superficie, una CPU, un accionador (para realizar los comandos de ajuste provenientes de la CPU), y un conjunto de cinco resortes. El accionador se conecta a los resortes mediante una interfaz mecánica. TecnoApretón recibe señales EMG asincrónicas de un electrodo de superficie y las pasa a la CPU. Cada EMG provoca una petición de interrupción, misma que la CPU tratará

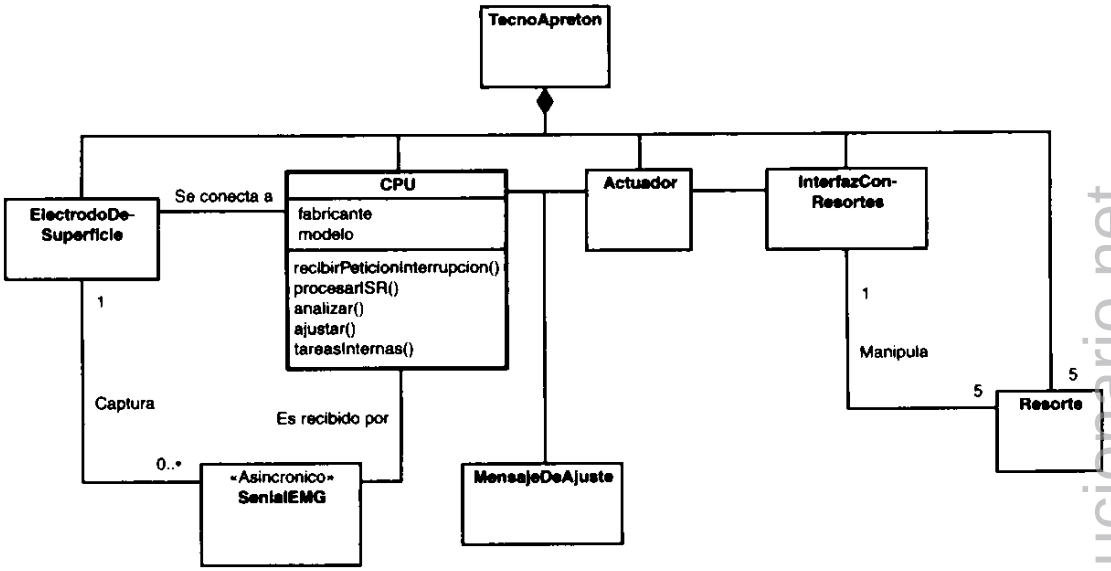
www.elsolucionario.net

con un ISR. El software en la CPU analiza las señales. Cuando el análisis está completo, la CPU envía una señal a un accionador para ajustar la tensión en los resortes. El actuador hace el ajuste mediante el manejo de la interfaz mecánica con los resortes.

La figura 23.6 muestra una estructura de clases que resume al párrafo anterior. Observe que el borde de la CPU está en negritas, lo que indica que es una clase activa. La idea es que la CPU siempre tendrá el control (en la recepción, análisis de señales y la administración de los ajustes). También realiza algunas tareas internas del sistema. Observe, a su vez, el uso de una clase de asociación para MensajeDeAjuste, que nos permite establecer parámetros al mensaje.

**FIGURA 23.6**

*Estructura de clases de TecnoApretón.*




SenialEMG y MensajeDeAjuste son importantes, así que nos enfocaremos en ellos. El sistema estará interesado en el momento en que llegue una señal y su potencia, por lo que horaDeLlegada y amplitud parecen ser atributos razonables para SenialEMG. A su vez, la SenialEMG tendrá indudablemente características complejas que están fuera del ámbito de este tema.

Para MensajeDeAjuste, tanto horaDeGeneracion y cantidadAjuste parecen ser atributos razonables.

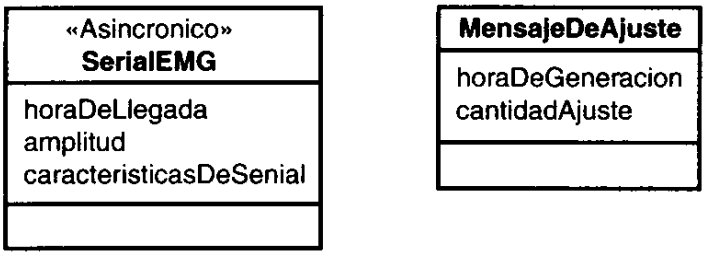
La figura 23.7 muestra los atributos de estas clases.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 23.7**

*Una mirada más cercana a SenialEMG y MensajeDeAjuste.*

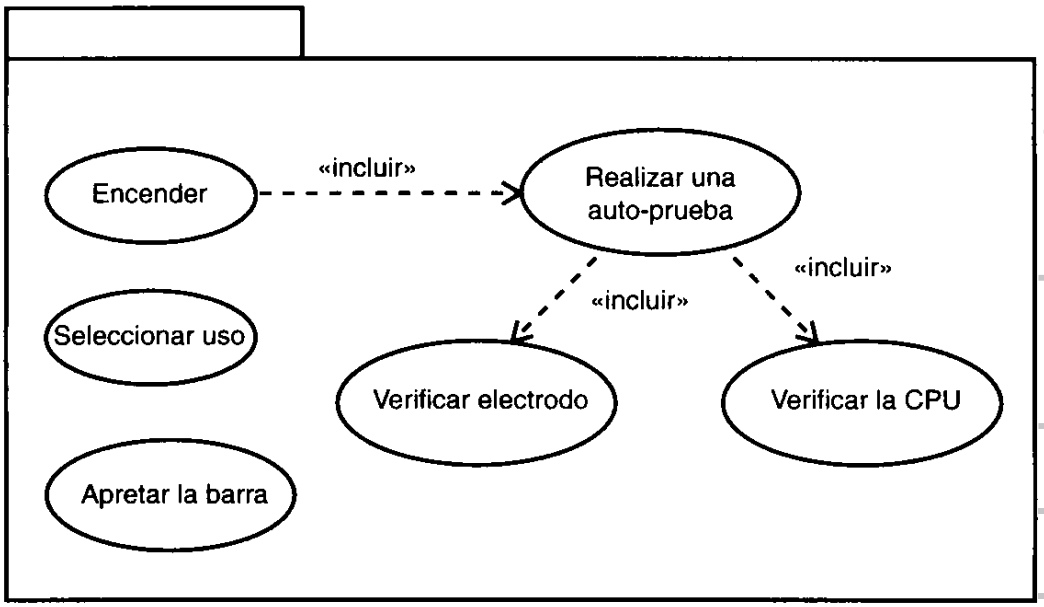



## Casos de uso

La sesión JAD que mencioné anteriormente (que dio por resultado la decisión de diseñar un sistema incrustado en lugar de una computadora de escritorio), también dio por resultado varios casos de uso, como se aprecia en la figura 23.8.

**FIGURA 23.8**

*Los casos de uso de TecnoApretón.*




Estos casos de uso determinan las facultades por generar en el sistema. "Encender" incluye a "Realizar una auto-prueba" la cual, a su vez, incluye "Verificar el electrodo" y "Verificar la CPU".

"Seleccionar uso" se refiere a diversas formas de ajustar el TecnoApretón, muchas de las cuales no se le ocurrieron al Sr. Nar cuando pensó en este dispositivo. Por ejemplo: los participantes de la JAD dijeron que les gustaría establecer repeticiones "negativas": una resistencia limitada cuando aprieten las barras, y una máxima cuando dejen de apretar.

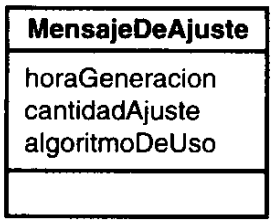
Esto significa que debemos agregar un atributo a MensajeDeAjuste para reflejar el uso del sistema. Podríamos llamarlo algoritmoDeUso y darle los valores posibles de aumentarTension y repNegativa. La figura 23.9 muestra la clase MensajeDeAjuste modificada.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 23.9**

*La clase MensajeDeAjuste modificada.*




## Interacciones

Ahora prestemos atención a "Apretar la barra", y asumamos que la persona ha seleccionado el modo originalmente concebido: incrementar la resistencia al incrementarse la actividad muscular. En esta parte del modelo, tenemos que asegurarnos de considerar restricciones de tiempo y cambios de estado. Asumamos que un intervalo de ciclos del reloj es de 20 microsegundos, y que el lapso desde que se recibe una señal hasta enviar un mensaje de ajuste no deberá ser mayor de 10 ciclos del reloj.

Una conjetura más: supongamos que el núcleo RTOS funciona de forma preferencial. Ello requiere de algunas pequeñas decisiones de modelado. Para empezar, y para reflejar la operación del núcleo, vamos a tratar las operaciones de analizar(), ajustar() y tareasInternas() de la CPU como subprocesos, y les asignaremos prioridades.

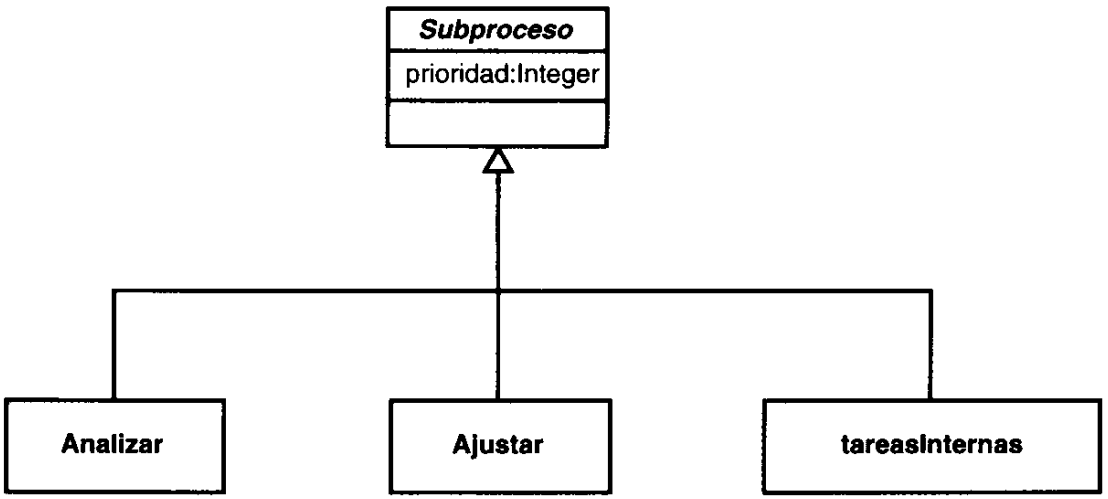
Para mostrarlas en nuestro modelo, tenemos que tratarlas como clases (algo que no solemos hacer con las operaciones). Este es un ejemplo de UML avanzado llamado *concreción*: tratar algo como si fuera una clase (u objeto) que no se suele tratar así. Cuando lo haga, enriquecerá a su modelo porque sus clases concretadas tendrán relaciones con otras clases, tendrán sus propios atributos y se convertirán en estructuras que podrá manipular y almacenar. En este caso, la concreción nos permite mostrar las prioridades de los subprocesos como atributos y utilizar los subprocesos en nuestros diagramas de interacción.

La figura 23.10 muestra la estructura de clases de los subprocesos de TecnoApretón.

www.elsolucionario.net

**FIGURA 23.10**

*Estructura de clases para los subprocesos de TecnoApretón.*




www.elsolucionario.net

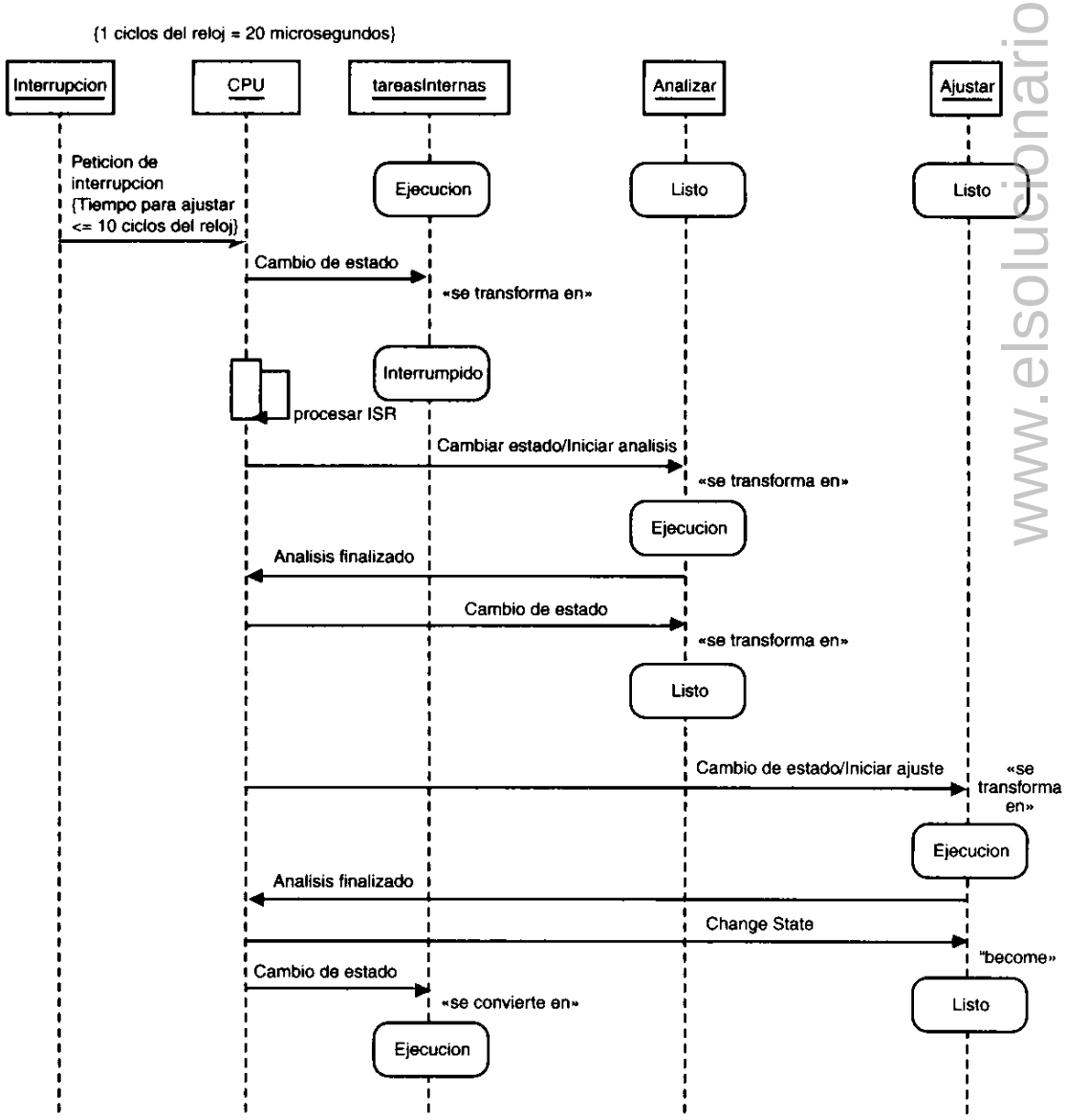
¿Cómo podríamos dar prioridades a nuestros subprocesos? Cuando llegue una petición de interrupción, la CPU tendrá que detener lo que hacía, guardar su contexto y atender la interrupción con un ISR. Nuestra operación procesarISR() toma la amplitud de SenialEMS y las otras características de señal compleja y las inserta en la memoria para analizar() para trabajar en las mismas. Entonces la operación analizar() debe tener la máxima prioridad. Seguirá la operación ajustar(). La operación tareasInternas() debe tener la menor prioridad.

He aquí un ejemplo de cómo todo esto funcionaría en conjunto de una manera preferencial. Si la CPU está a la mitad de la ejecución de algunas operaciones internas y llega una señal, interrumpirá lo que esté haciendo. La CPU ejecutará procesarISR() y obtendrá los valores adecuados de la señal. ¿Qué ocurre después? Regresar a las tareas internas no sería productivo. En vez de ello, la CPU ejecutará la operación de mayor prioridad, analizar(), seguida por ajustar().

La figura 23.11 le muestra el diagrama de secuencias para "Apretar la barra". La figura 23.12 le muestra un diagrama de colaboraciones para este caso de uso. En ambos diagramas utilizamos llaves para indicar restricciones de tiempo (medidas por ciclos del reloj).

**FIGURA 23.11**

*Diagrama de secuencias para "Apretar la barra".*

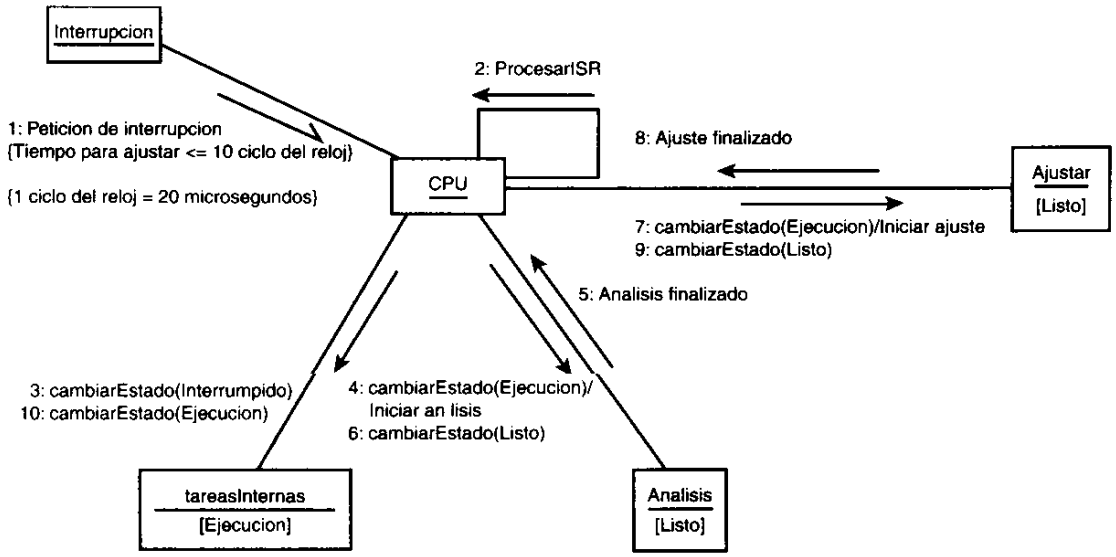



www.elsolucionario.net

www.elsolucionario.net

**FIGURA 23.12**

*Diagrama de colaboraciones para "Apretar la barra".*

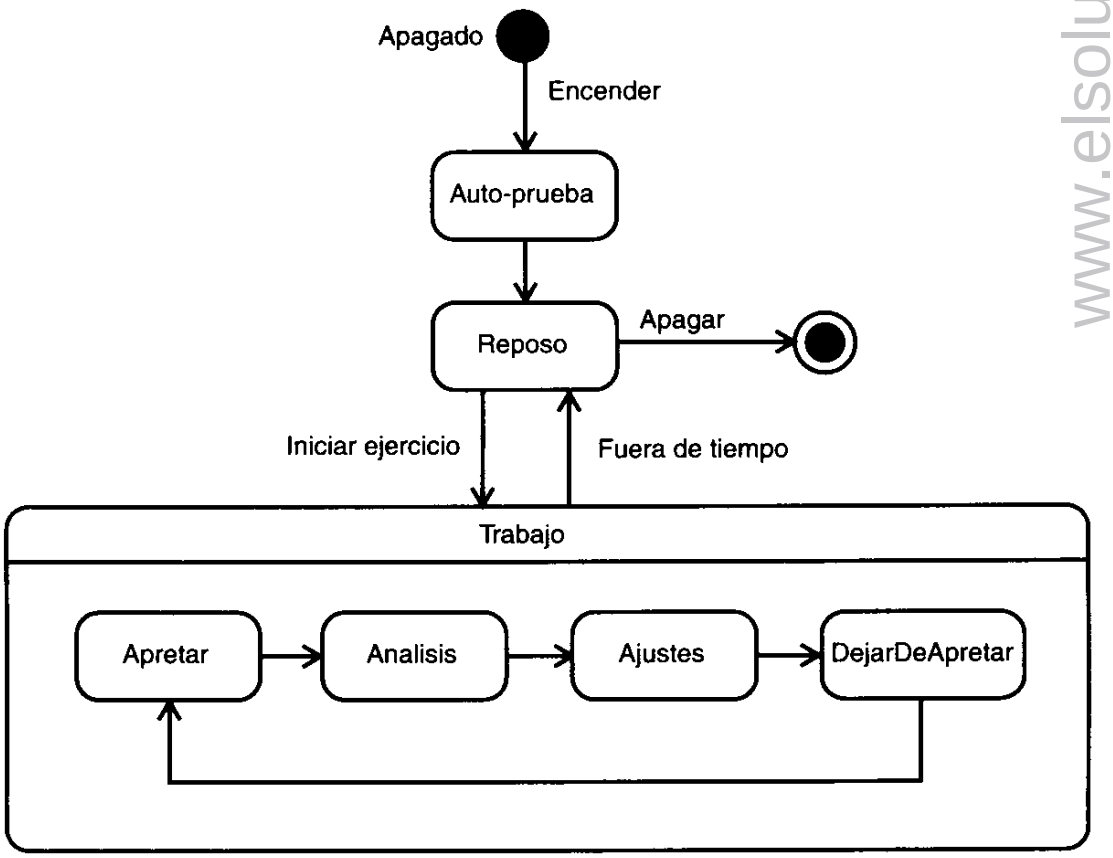



## Cambios de estado generales

Además de los cambios de estado dentro de una interacción, podemos examinar los cambios de estado en todo el sistema. Por lo general, esperamos que TecnoApretón esté en modo de Trabajo o en modo de Reposo (por ejemplo, entre lapsos de un ejercicio). También podría estar en el estado de Apagado. Como podría imaginar, el estado Trabajo es un objeto compuesto. La figura 23.13 muestra los detalles.

**FIGURA 23.13**

*Cambios de estado de TecnoApretón.*




www.elsolucionario.net

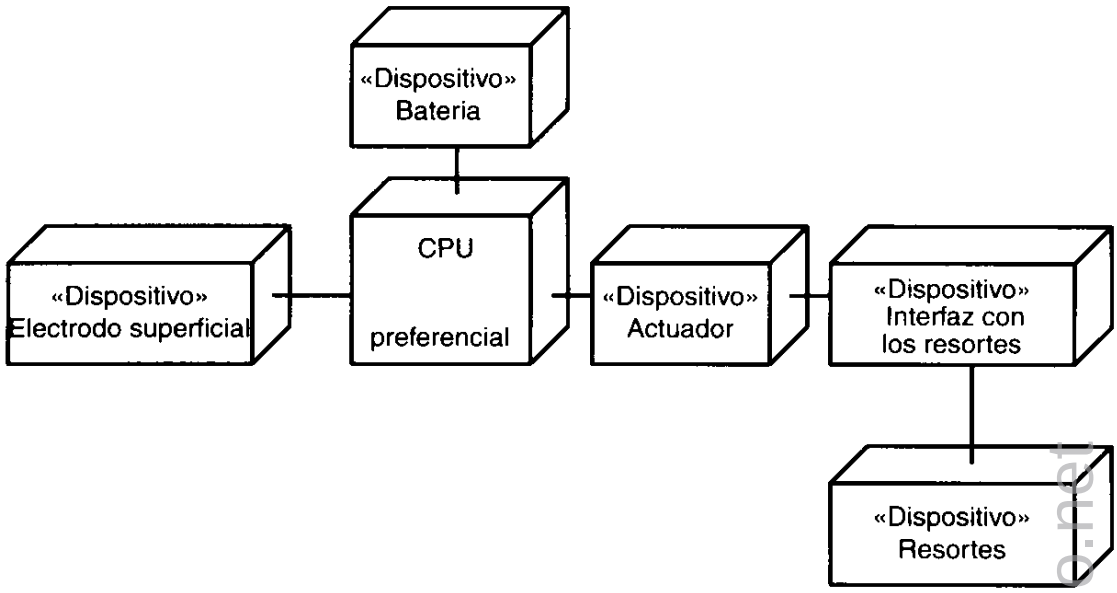
www.elsolucionario.net

## Distribución

¿Qué tal luciría TecnoApretón una vez que se lleve a cabo? La figura 23.14 es un diagrama de distribución que muestra las partes del sistema, junto con una batería que suministra la energía.

**FIGURA 23.14**

*Un diagrama de distribución para TecnoApretón.*




# Flexiones en sus músculos

Cuando los socios recibieron los diagramas UML de TecnoApretón, empezaron las maquinaciones.

"Éste es un concepto que podríamos expandir", dijo Goniff.

"¿Cómo?", preguntó Nar.

"Piénsenlo. ¿Cuántos músculos hay en el cuerpo humano? Podríamos crear un dispositivo ejercitador inteligente que se centre en varios de ellos."

"Ah, ¿sí?" preguntó nuevamente Nar, cautivado.

"¡Claro!", dijo Goniff. "Si llevamos el concepto de resortes, electrodos y CPU un poco más allá, podríamos desarrollar una barra con pesas portátil e inteligente que la gente podría llevar consigo cuando viajara. No pesaría mucho, dada la liviandad de los resortes que darían la resistencia y la CPU que daría la inteligencia. Podríamos llamarlo: 'TecnoPesas'."

"Sí", dijo Nar, "o podríamos seguir otro derrotero y hacer máquinas por separado para cada parte del cuerpo".

"¡Así es! Algo así como 'TecnoPecho'."

"O 'TecnoBrazo'".

"O 'TecnoPierna'."

www.elsolucionario.net

www.elsolucionario.net

"¿Qué tal 'TecnoAsentadera'?"

En este momento, LaHudra ya no podía soportar más.

"Tengo uno para ustedes dos", les dijo a sus socios.

"¿Cuál?" preguntaron al unísono.

"TecNoSePasen."

# Resumen

Un sistema incrustado es una computadora que se encuentra dentro de algún otro dispositivo, como un electrodoméstico. La programación de un sistema incrustado requiere mucho conocimiento del dispositivo donde se encontrará el sistema. Un sistema incrustado podría ser *tolerante*, significa que no tiene que cumplir con plazos rigurosos, o *estricto*, que sí tiene que cumplir con ellos.

El tiempo, los subprocesos (programas sencillos que son partes de una aplicación) y las interrupciones (dispositivos de hardware que dejan saber a la CPU que ha ocurrido algo) son conceptos importantes en los sistemas incrustados. Un interruptor en particular, el ciclo del reloj, sucede a intervalos regulares y funciona como un latido del corazón del sistema.

Un sistema operativo en tiempo real (RTOS) dirige el tráfico entre subprocesos e interrupciones. El núcleo es la parte del RTOS que administra el tiempo que ocupa la CPU en subprocesos individuales. El programador de tiempo del núcleo determina cuál tarea se ejecutará a continuación. Un núcleo puede ser preferencial (en el que un subproceso de mayor prioridad tiene preferencia sobre uno de baja prioridad cuando finaliza una rutina para el servicio de interrupciones) o cooperativo (en el que la tarea interrumpida continúa después que la rutina para el servicio de interrupciones finaliza).

Estos conceptos los aplicamos mediante el modelado de un inteligente dispositivo ejercitador que varía su resistencia en función de la fuerza con que trabaje un músculo.

# Preguntas y respuestas

**P Usted habló de sistemas "inteligentes". ¿Tales sistemas incrustados incluyen algo como la Inteligencia Artificial?**

**R** Así es. Una variedad de la IA, llamada "lógica vagamente definida" o "fuzzy logic", se encuentra en el corazón de diversos tipos de sistemas incrustados.

**P ¿Un tipo de RTOS es más adecuado que otro para cierto tipo de aplicaciones de sistemas incrustados?**

**R** Sí. Un tipo del cual no di mayores explicaciones, el superciclo, es el RTOS más sencillo. Con frecuencia está incrustado en aplicaciones de alto volumen como los juguetes. El núcleo preferencial es el RTOS elegido para sistemas complejos.

www.elsolucionario.net

www.elsolucionario.net

# Taller

He incrustado algunas preguntas para verificar su nuevo conocimiento, y las respuestas están incrustadas en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿Qué es un sistema incrustado?
2. ¿Qué es un evento asincrónico?
3. En términos de sistemas incrustados, ¿qué es un sistema "estricto" y qué un sistema "tolerante"?
4. ¿Qué sucede en un "núcleo preferencial"?

## Ejercicios

1. Imagínese un sistema incrustado para un tostador. Asuma que el tostador cuenta con un sensor que analiza una rebanada de pan y su nivel de tueste. Asuma, a su vez, que puede establecer qué tan tostado desea su pan. Dibuje un diagrama de clases para este sistema. Incluya al sensor, la CPU y el elemento calentador (¡y la rebanada de pan!).
2. Dibuje un diagrama de secuencias para el sistema incrustado del tostador. Justifique su elección de un núcleo preferencial o uno cooperativo. Y, aprovechando el viaje, dibuje también un diagrama de distribución.

www.elsolucionario.net

www.elsolucionario.net

# HORA 24

# El futuro del UML

Por último, analizaremos algunas extensiones actuales del UML y algunas extensiones potenciales.

En esta hora se tratarán los siguientes temas:

- Extensiones para los negocios
- Lecciones de las extensiones de negocios
- Interfaces gráficas de usuario
- Sistemas expertos

Hemos llegado a la última hora. Ha sido un largo trecho, pero en el proceso hemos visto mucho del UML. En las últimas dos horas ha visto aplicaciones en áreas innovadoras. En esta hora, veremos un panorama de todo con una extensión actual UML y una mirada a otras áreas de aplicación del UML.

Hemos tratado las extensiones del UML en la hora 14, "Nociones de los fundamentos del UML". Como lo indicamos entonces, puede extender el UML mediante la adición de estereotipos que le permitan hacer su propio

www.elsolucionario.net

www.elsolucionario.net

modelo de su dominio. También podrá agregar estereotipos gráficos que aclaren la información trasladada a su modelo. Los diagramas de distribución son buenos ejemplos de ello porque las figuras con frecuencia sustituyen a los iconos cúbicos del UML.

La meta de esta hora es que empiece a pensar en cómo podría aplicar el UML en su dominio. Como cualquier lenguaje, el UML se encuentra en proceso de evolución. Su futuro depende de cómo lo utilicen y lo extiendan los modeladores.

# Extensiones para los negocios

Una extensión popular es un conjunto de estereotipos diseñados para modelar un negocio. Los estereotipos abstraen ciertas ideas primordiales de las que se conforma un negocio. Podrá visualizarlas en términos de símbolos UML que ya conoce o como iconos especializados (creados por el Amigo del UML Ivar Jacobson). La intención es la de modelar situaciones del mundo de los negocios, en lugar de que funcionen como los fundamentos para la construcción del software.

Dentro de un negocio, la clase obvia es *trabajador*. En el contexto de esta extensión UML, un trabajador actúa dentro de los negocios, interactúa con otros trabajadores y participa en los casos de uso. Un trabajador puede ser un *trabajador interno* o un *trabajador eventual*. Un trabajador interno interactúa con otros trabajadores dentro de los negocios, y un eventual lo hace con los actores fuera de los negocios. Una *entidad* no inicia ninguna interacción, pero participa en los casos de uso. Los trabajadores interactúan con las entidades.

TERMINO NUEVO

La figura 24.1 le muestra la acostumbrada notación UML para estos estereotipos, junto con los iconos especializados. Para cada uno, he incluido un ejemplo del dominio restaurante.

Las extensiones de negocios incluyen a dos estereotipos de asociación: «comunica» y «suscribe». El primero es para las interacciones entre objetos. El segundo describe una asociación entre un origen (llamado *suscriptor*) y un destinatario (llamado *publicador*). El origen establece un conjunto de eventos. Cuando uno de esos eventos sucede en el destinatario, el origen recibe una notificación.

TÉRMINO NUEVO

Las entidades se combinan para conformar *unidades de trabajo*, que son conjuntos de objetos orientados a tareas. Las unidades de trabajo, las clases y las asociaciones se combinan para conformar *unidades organizativas*. Una unidad organizativa (que puede incluir a otras unidades organizativas) corresponde a una unidad organizativa del negocio.

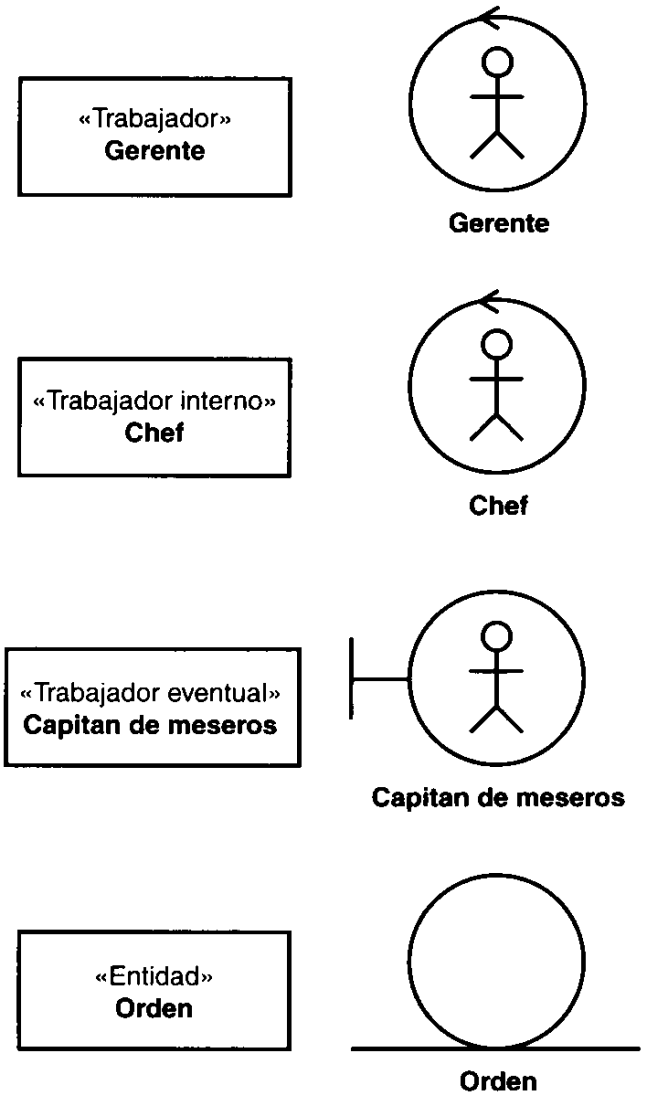
TÉRMINO NUEVO

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 24.1**

*Estereotipos para el modelado de negocios.*




# Lecciones de las extensiones de negocios

Las extensiones de negocios nos enseñan algunas lecciones importantes. Para empezar, es evidente que con algo de imaginación es posible utilizar iconos sencillos y representaciones que capturen aspectos fundamentales del dominio. El mundo operativo es "sencillo". En segundo término, las representaciones nos ayudan a pensar respecto a, y a crear soluciones en, un dominio.

Tomaremos en cuenta tales lecciones conforme exploremos y llevemos al UML a dos importantes proyectos de modelado: las interfaces gráficas de usuario y los sistemas expertos.

# Interfaces gráficas de usuario

Un sello de los paquetes contemporáneos de software, la GUI (interfaz gráfica de usuario) ha llegado para quedarse. GRAPPLE y otros procesos y metodologías de desarrollo se aplican a una sesión JAD para la creación de la GUI de una aplicación.

www.elsolucionario.net

www.elsolucionario.net

En un documento de diseño, por lo general, incluirá capturas de pantalla para mostrar a sus clientes y desarrolladores cómo lucirá la GUI a los usuarios. Por varias razones, aun podría necesitar un diagrama especializado para modelar una GUI.

## Conexiones a casos de uso

La razón primordial tiene que ver con los casos de uso. Como la mayor parte de un proyecto de desarrollo, el desarrollo de la GUI está conducido por casos de uso. De hecho, la GUI se conecta directamente con los casos de uso porque es la ventana por la cual el usuario final iniciará y completará los casos de uso. Podría ser difícil utilizar capturas de pantalla para establecer la relación entre las pantallas y los casos de uso.

Otra razón es que probablemente necesitemos capturar la evolución en el proceso de razonamiento conforme la GUI tome forma. En GRAPPLE, el desarrollo de la GUI inicia cuando los usuarios finales que participen en la sesión JAD maniobren las notas autoadheribles (que representan controles en la pantalla) en grandes hojas de papel (que representan pantallas). Podría ser útil contar con un tipo de diagrama que capture los resultados de tales maniobras de manera directa (uno que un modelador pueda modificar con facilidad, cuando los participantes de la sesión JAD modifiquen el diseño).

Un diagrama que muestre las conexiones de las pantallas con los casos de uso ayudaría a los que participan en la sesión JAD a recordar lo que se supone que hará cada pantalla cuando coloquen los componentes de la misma. Mostrar las conexiones con los casos de uso también ayudará a asegurar que todos los casos de uso sean instaurados en el diseño final.

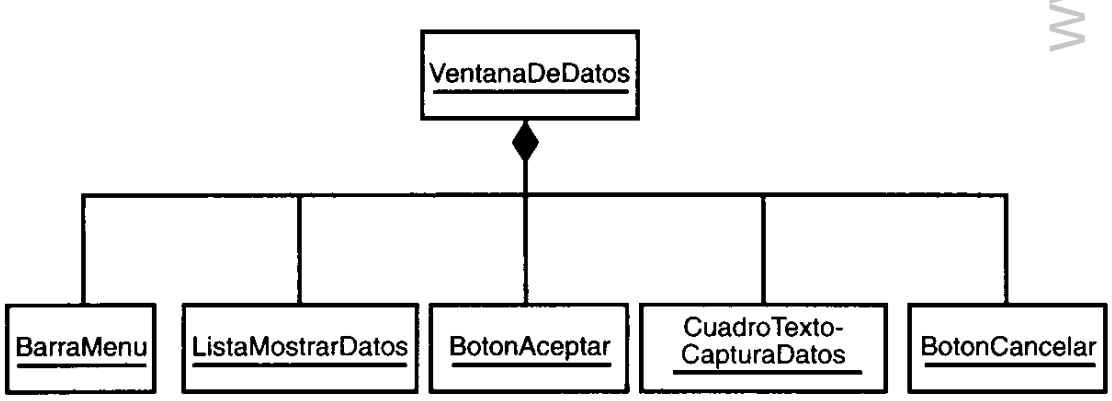
## Modelado de la GUI

Un modelo típico en UML podría representar a una ventana de una aplicación particular como un objeto compuesto de varios controles, como se ve en la figura 24.2.

www.elsolucionario.net

**FIGURA 24.2**

*Un modelo UML de una ventana.*

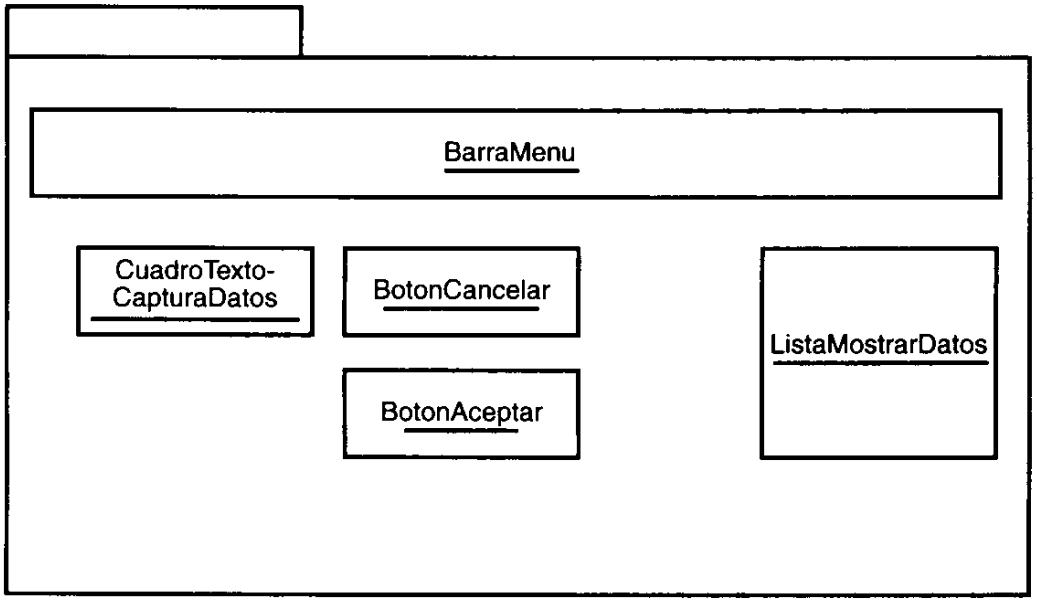



Podríamos utilizar atributos para agregar una ubicación en el espacio de cada componente: una ubicación horizontal y una vertical medida en píxeles. Otro par de atributos podría representar al tamaño del componente (alto y ancho). No obstante, es más fácil abarcar ambos componentes si los visualizamos. Podríamos especificar que un paquete represen-

tará a una ventana, y que la ubicación y tamaño de los objetos dentro del paquete refleje su ubicación y tamaño en la ventana. La figura 24.3 hace lo que indiqué.

**FIGURA 24.3**

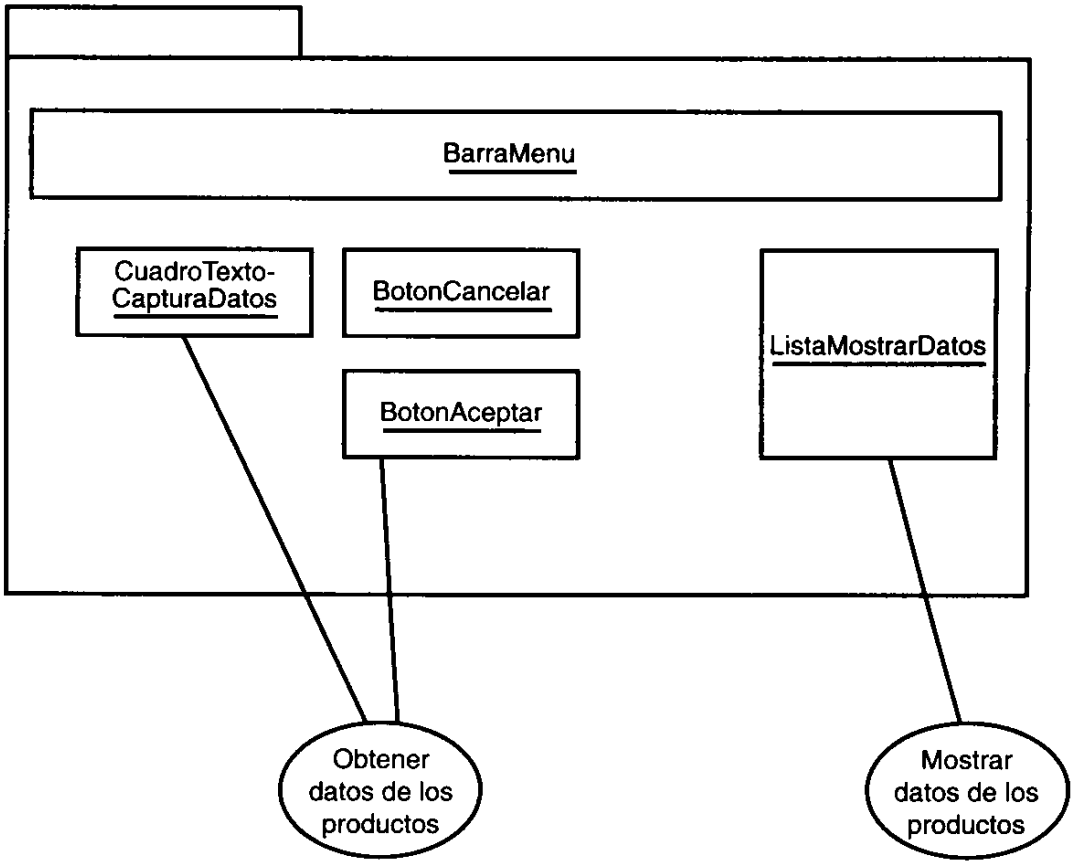
*Modelo de una ventana que muestra la ubicación de los componentes.*




La figura 24.4 es el diagrama híbrido que agrega el toque final al mostrar las conexiones con los casos de uso.

**FIGURA 24.4**

*Modelado de una ventana que muestra la forma en que los componentes de la pantalla se conectan a los casos de uso.*




www.elsolucionario.net

www.elsolucionario.net

Este tipo de modelado no evita que se muestren las capturas de pantalla. En lugar de ello, podría ser una adición útil: un diagrama esquemático que se centre en el panorama general.

# Sistemas expertos

Los sistemas expertos surgieron a la popularidad en la década de los años ochenta. Cuando aparecieron no fueron más que una curiosidad, pero hoy son parte de la corriente principal del cómputo.

Un sistema experto está diseñado para capturar el conocimiento y experiencia de una persona versada en un dominio específico. Almacena esa experiencia en un programa de cómputo. La intención es utilizar al sistema experto para responder a preguntas repetitivas para que la persona experta no tenga que hacerlo, o almacenar la experiencia para que esté disponible cuando la persona no lo esté.

## Componentes de un sistema experto

TERMINO NUEVO

La experiencia se encuentra en la *base de conocimientos* del sistema experto, como un conjunto de reglas condicionales *if-then* (si-entonces). La parte *if* de la condición describe ciertas situaciones reales en el dominio del experto. La parte *then* establece las acciones por realizar en tal situación. ¿De qué manera se registra la experiencia en la base de conocimientos? Un *ingeniero de conocimiento* realiza extensas entrevistas con un experto, graba los resultados y los presenta en software. Es similar a la entrevista que se realiza en un análisis de dominio, aunque las sesiones para la ingeniería del conocimiento son, por lo general, más extensas.

La base de conocimientos no es el único componente en un sistema experto. Si lo fuera, un sistema experto podría ser tan sólo una lista de reglas condicionales if-then. Lo que se necesita es un mecanismo para operar a lo largo de la base de conocimientos para resolver un problema. A tal mecanismo se le conoce como *motor de deducciones*. Otra pieza necesaria del rompecabezas es un *área de trabajo* que contenga las condiciones de un problema que el sistema tenga que resolver. Claro está que otro componente más es la interfaz del usuario para indicar las condiciones del problema. La captura de condiciones podría realizarse mediante una lista de verificación, preguntas y respuestas de opción múltiple o un sistema extremadamente sofisticado sentado en el idioma natural. La figura 24.5 le muestra un diagrama de clases UML de un sistema experto.

Para interactuar con un sistema experto, el usuario captura las condiciones de un problema en la interfaz del usuario, misma que las almacena en el área de trabajo. El motor de deducciones se vale de tales condiciones para buscar una solución en la base de conocimientos. La figura 24.6 presenta un diagrama de secuencias para este proceso.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 24.5**

*Diagrama de clases de un sistema experto.*

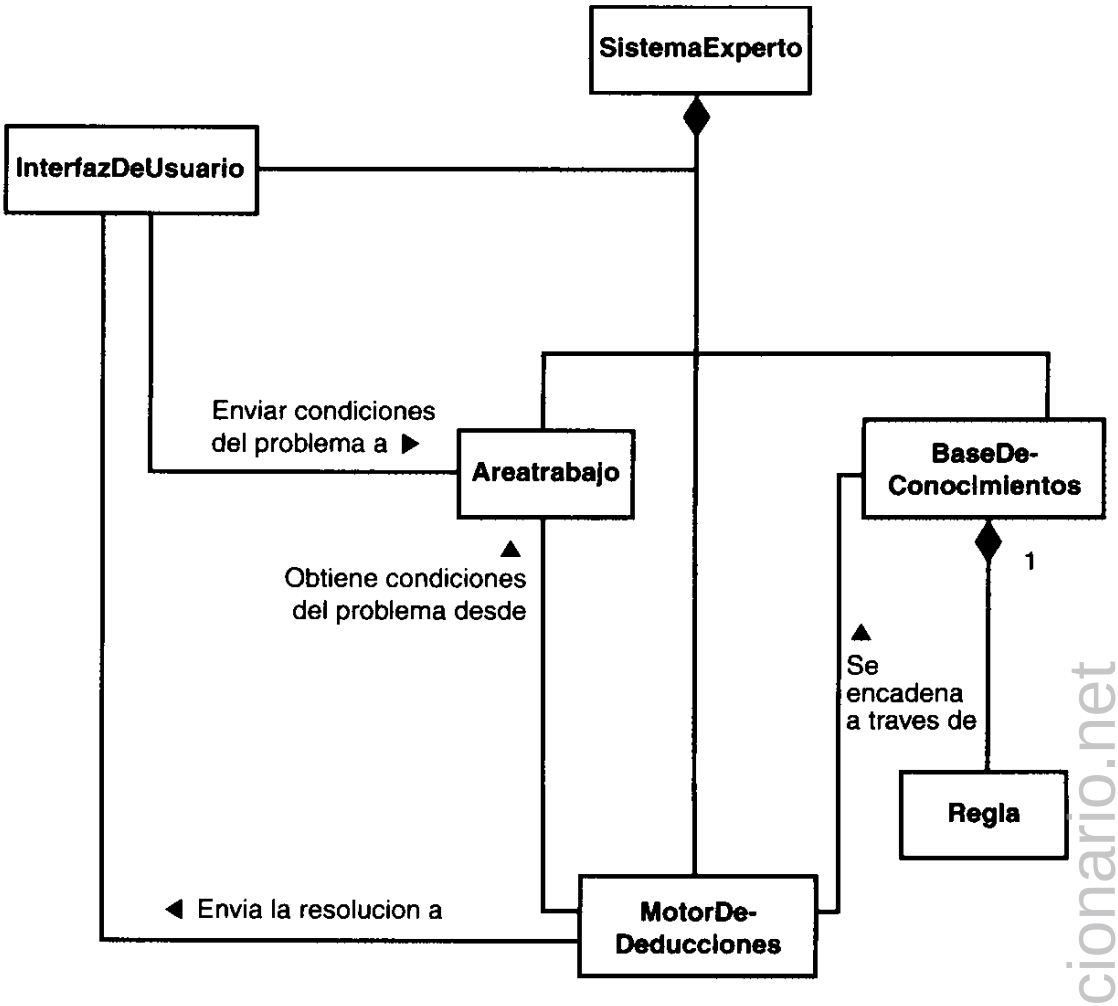
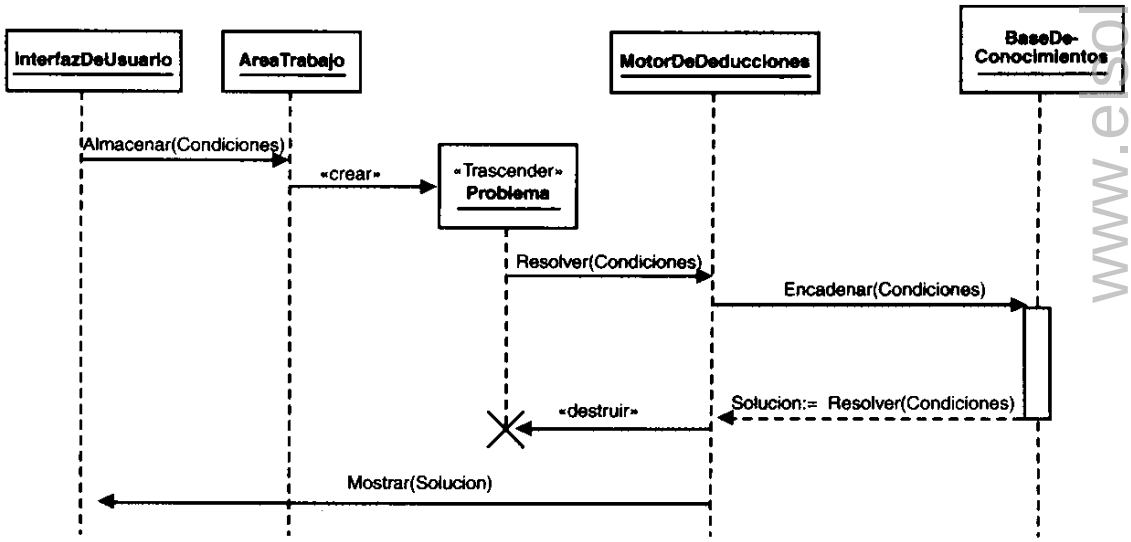



**FIGURA 24.6**

*Interacción con un sistema experto.*




www.elsolucionario.net

El área de trabajo podría equipararse a la memoria temporal (datos en caché), en tanto que la base de conocimientos sería la memoria permanente (datos en disco) y el motor de deducciones sería el proceso para resolver el problema.

## Un ejemplo

Un motor de deducciones por lo general analiza su base de conocimientos en una de dos formas, y la mejor manera de explicarlo es mediante un ejemplo. Suponga que tenemos un sistema experto que captura la experiencia de un fontanero. Si uno de sus grifos gotea, utilizaría al sistema experto y le explicaría el problema. El motor de deducciones haría el resto.

Dos de las reglas en su base de conocimientos podrían lucir así:

Regla 1:

SI tiene un grifo que gotea

Y el grifo es de compuerta

Y la gotera está en la manija

ENTONCES ajuste el empaque

Regla 2:

SI el empaque está ajustado

Y la gotera persiste

ENTONCES reemplace el empaque

Sin profundizar en el mundo de la fontanería, bastará decir que estas dos reglas están relacionadas (obviamente): observe la similitud entre la parte then de la Regla 1 y la parte if de la Regla 2. Tal similitud es el fundamento para analizar la base de conocimientos (la cual tiene mucho más que dos reglas). El motor de referencia podría iniciar con una solución potencial como "reemplazar el empaque" en la Regla 2, e ir en retrospectiva para ver los detalles del problema que demanda tal solución.

¿De qué manera el motor de deducciones "va en retrospectiva"? Analiza la parte if de la regla que tenga la solución y busca una regla cuya parte then concuerde. En nuestro ejemplo de dos reglas, eso es fácil: la Regla 1 tiene una parte then coincidente. En las aplicaciones de tono empresarial, esto no es tan sencillo debido a que una base de conocimientos podría almacenar cientos, tal vez miles, de reglas.

www.elsolucionario.net

www.elsolucionario.net

Después de que el motor de deducciones haya encontrado una regla, verifica si las condiciones de la parte if concuerdan con las condiciones del problema. Si es así, el motor continúa su búsqueda en la misma dirección: una parte then, verifica la parte if, otra parte then que concuerde y así sucesivamente. Cuando el motor de deducciones se queda sin reglas, le pide más información al usuario. La meta de ello es que si el trayecto por las reglas es exitoso (esto es, que concuerde con las condiciones del problema), el sistema ofrecerá la solución potencial original como la solución al problema. Si no es el caso, el sistema intentará un nuevo trayecto.

TERMINO NUEVO

Esta técnica de probar una solución y verificar si las condiciones asociadas concuerdan con las condiciones del problema se conoce como *encadenado regresivo*; "regresivo" porque inicia con las partes then y procede a examinar las partes if.

TERMINO NUEVO

Como podrá imaginar, otra técnica inicia con las partes if y corresponde con las partes then, y se conoce como *encadenado progresivo*. Así es como trabaja. El usuario establece las condiciones del problema y el motor de deducciones localiza una regla cuya parte if concuerde con las condiciones. Verifica la parte then y busca una regla cuya parte if concuerde con tal parte then. En nuestro ejemplo, suponga que la parte if de la Regla 1 concuerde con las condiciones del problema. El motor de deducciones verifica la parte then de la Regla 1 y, luego, busca una regla con una parte if que concuerde. Nuevamente, esto es sencillo en nuestro caso con sólo dos reglas. Cuando el sistema se queda sin reglas por comparar, ofrece las parte then de la regla final como la solución. La parte "progresiva" del "encadenado progresivo" se refiere a este movimiento de las partes if a las partes then.

Si tuviéramos que modelar un sistema experto como en la figura 24.5, sería de mucha ayuda agregar un estereotipo que indique el tipo de encadenamiento que realice el motor de deducciones. Agregaríamos ya sea «encadenamiento progresivo» o «encadenamiento regresivo» a la clase compuesta SistemaExperto.

Ambos tipos de encadenamiento son ejemplos del patrón de diseño Cadena de responsabilidades que vio anteriormente. En cada uno, el sistema busca a un "sucesor" de la regla.

Así como en ocasiones la Cadena de responsabilidades finaliza sin encontrar a un sucesor, un sistema experto tampoco le presentará siempre una solución.

## Modelado de la base de conocimientos

¿Qué es lo que puede agregar el UML a todo esto, y para qué lo querríamos? Uno de los puntos primordiales en el desarrollo de un sistema experto es la ausencia de una norma sólida para generar una representación visual de las reglas de la base de conocimientos. Una representación basada en el UML podría requerir de un largo proceso para

www.elsolucionario.net

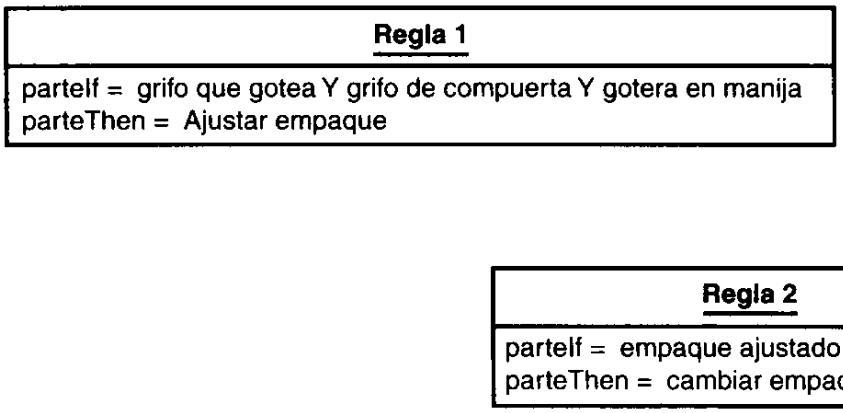
www.elsolucionario.net

estandarizar el campo que propenda a buenas prácticas de documentación. No es suficiente que el conocimiento se encuentre en una representación informática dentro de una base de conocimientos: las reglas deberían estar también en un documento.

¿Cómo podríamos representar a una base de conocimientos bajo el espíritu del UML? Es evidente que una posibilidad sería la de representar cada regla como un objeto. Contar con uno de los atributos para que sea la parte if, otro la parte then y agregar otros atributos conforme sea necesario. La figura 24.7 le muestra este diseño.

**FIGURA 24.7**

*Representación de las reglas como objetos.*




Aunque esto es eminentemente factible (y muchos desarrolladores lo hacen), creo que las reglas son muy importantes para garantizar su propia representación. Además de ser el fundamento de las bases de conocimientos, el creciente énfasis en la administración del conocimiento dentro de las organizaciones e instituciones espera algo único.

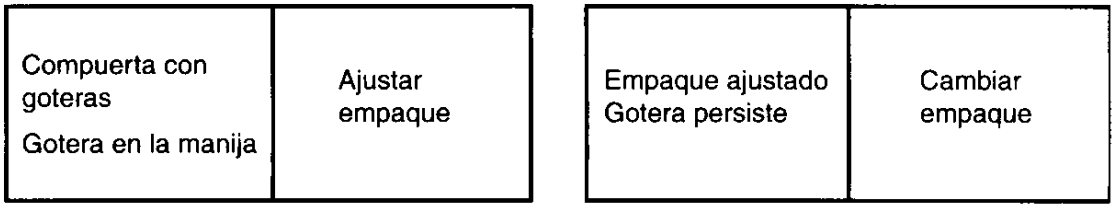
¿Cómo podría lucir tal representación única? Para empezar, necesitamos asegurarnos que mostraremos algo que dé el contenido de la parte if de la regla, y el de la parte then. Para hacer útil a esta representación, también necesitamos algo para visualizar las conexiones entre las reglas.

Esto podría dificultarse con rapidez. Las reglas de talla industrial tienden a contar con mayor información que las dos reglas de fontanería que le mostré, y las reglas se inclinan a proliferar. Tenemos que balancear la proliferación respecto a la necesidad de la simplicidad.

Primero cree un sencillo icono que represente a la regla. Empezaremos con un cuadro dividido a la mitad por una línea vertical. La mitad izquierda representará a la parte if y la derecha a la parte then. Dentro de cada parte escribiremos un resumen significativo del contenido. La figura 24.8 le muestra lo que le quiero decir, con el ejemplo de las dos reglas de fontanería.

**FIGURA 24.8**

*Las dos reglas de fontanería moldeadas en una representación visual.*

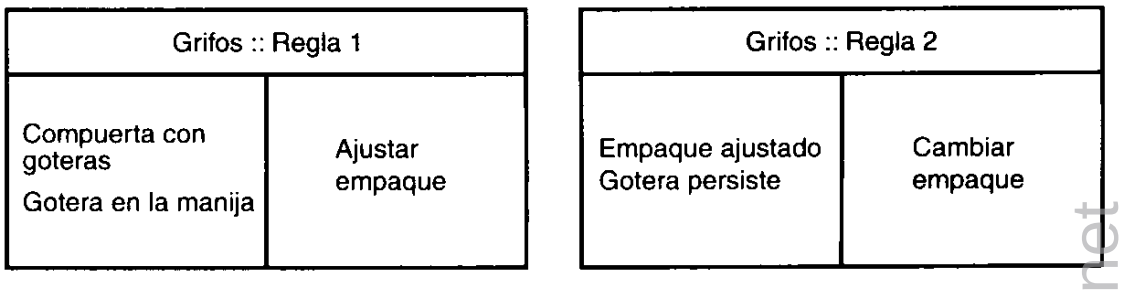



www.elsolucionario.net

Ahora tenemos que incorporar cierta información de identificación para cada regla. Agreguemos un compartimiento en la parte superior de cada cuadro que contenga un identificador numérico. Esto logra un par de cosas: (1) Convierte a cada regla en única, y (2) muestra a dónde ir en un catálogo de reglas para obtener una descripción y explicación completa de la regla. Si una regla es parte de un subgrupo de reglas (como en un subconjunto "grifo" de nuestra base de conocimiento de fontanería), podemos tratar al subgrupo como un paquete. Luego, agregar el paquete de información al identificador de la forma usual propia del UML: hacer que el nombre del paquete anteceda a un par de dos puntos (::), que antecedan al identificador. La figura 24.9 muestra tal adición.

**FIGURA 24.9**

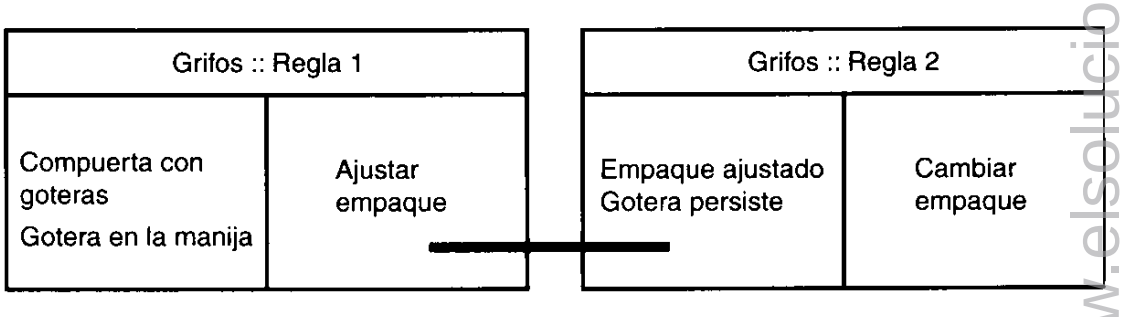
*Adición de un identificador a cada regla.*




Ahora, representaremos la relación entre los dos como una línea entre la parte then de la Regla 1 y la parte if de la Regla 2. La figura 24.10 le muestra la conexión.

**FIGURA 24.10**

*Conexión de la parte then de una regla a la parte if de la otra.*




A diferencia de los dos conjuntos de pares de reglas, una regla en un sistema experto verdadero se relaciona usualmente a más de una regla distinta. Si las reglas relacionadas no están próximas —ya sea en la base de conocimientos o en la documentación— será útil tener una forma de mostrar la relación aún cuando no podamos contar con líneas de conexión.

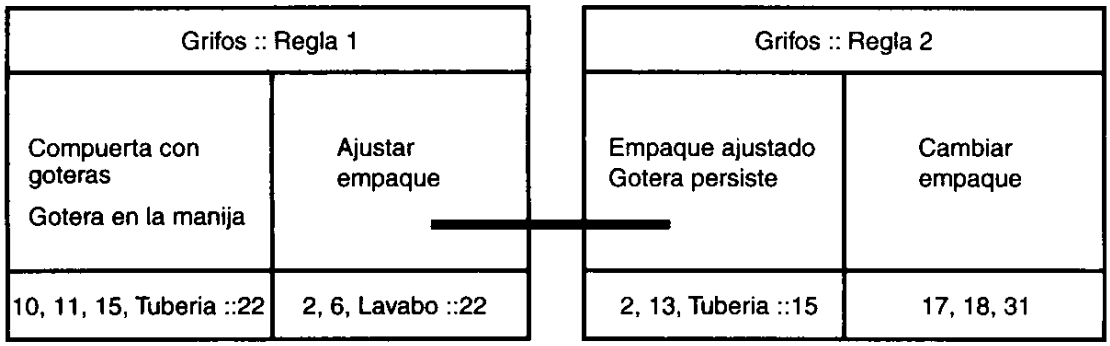
Los compartimientos de la parte baja del icono nos permitirán lograrlo. Si los colocamos por debajo de los que ya tenemos, podremos mostrar identificadores para otras reglas, como en la figura 24.11. El compartimiento inferior de la izquierda identifica a las reglas cuyas partes then se conectan con la parte if de ésta. El compartimiento inferior de la derecha identifica reglas cuyas partes if se conectan con la parte then de ésta.

www.elsolucionario.net

www.elsolucionario.net

**FIGURA 24.11**

*Los compartimientos en la parte inferior del icono de la regla identifican a las reglas relacionadas.*




Como en el caso de los diagramas de clases, los compartimientos dentro del icono de la regla podrían ser omitidos de acuerdo con el propósito del diagrama. La idea es mostrar de manera concisa las conexiones entre las reglas, así como su contenido, y con ello comunicar con claridad la naturaleza de la base de conocimientos.

El modelo del sistema experto es más "drástico" que el de la GUI, en el sentido de que propone un nuevo "elemento de perspectiva" (el icono de regla) para el UML. El modelo para la GUI, por otra parte, es un diagrama híbrido que consta de elementos UML actuales.

# Eso es todo, amigos

Hemos llegado al final del camino. Ahora que cuenta con todo un equipaje lleno de trucos del UML, ya está listo para continuar por sí mismo y aplicarlos a su dominio. Verá que conforme obtenga experiencia, hará adiciones a tal equipaje. Posiblemente tenga algunas sugerencias por agregar al UML. Si es su caso, continuará con una gran tradición.

A principios del siglo XX, el renombrado matemático Alfred North Whitehead apuntó la importancia de los símbolos y de su uso. Un símbolo, como indicó, representa a una idea: la importancia de un símbolo es que rápida y concisamente muestra la forma en que una idea encaja en un complejo grupo de ideas diversas.

Al principio de un nuevo milenio, las observaciones de Whitehead aún se aplican en el mundo del desarrollo de sistemas. Los símbolos cuidadosamente diseñados nos muestran el proceso del raciocinio y las complejidades que hay detrás de los maravillosos sistemas que intentamos generar, y nos ayudan a asegurar su rendimiento eficiente cuando los generemos.

# Resumen

Conforme los modeladores extiendan y moldeen al UML para cumplir con sus necesidades, modelarán su propio futuro. En esta hora, hemos visto una extensión para el modelado de los negocios y sugerí algunas formas de aplicar el UML en otras áreas.

www.elsolucionario.net

www.elsolucionario.net

Como lección de la simplicidad de las extensiones de negocios, hemos explorado las formas de modelar las GUI y los sistemas expertos. Para modelar una GUI, establecemos un diagrama híbrido que muestre las relaciones de espacio de los componentes de la pantalla, y que muestre sus conexiones y casos de uso. Esto tiene la ventaja de mostrar la evolución de una GUI conforme toma forma, y mantiene a los casos de uso correspondientes en el centro de la atención.

En un sistema experto, las reglas de condiciones (if-then) son el bloque de construcción de una base de conocimientos, el componente que contiene el conocimiento de un experto en algún dominio humano. Sugerimos un diagrama que visualice las reglas y sus relaciones internas. En este diagrama, un cuadro dividido en compartimientos modela a la regla. Un compartimiento contiene al identificador de la regla, otro resume la parte if, otro la parte then y otras dos muestran las reglas relacionadas. Los vínculos a las reglas adyacentes aparecen como líneas de conexión entre las partes adecuadas de las reglas.

# Preguntas y respuestas

**P Aunque en principio parecería que un sistema experto no es difícil de modelar, parece que podría convertirse en un programa extremadamente difícil de escribir.**

**R** Lo será si tiene que crear uno desde el inicio. Por fortuna, la mayor parte de la programación la realiza usted en un paquete llamado *shell* para un sistema experto. Todos los componentes ya están hechos. Tan sólo deberá agregar el conocimiento. No obstante, extraer el conocimiento de una persona experta no siempre es una tarea fácil.

**P ¿Qué los vendedores de shells para sistemas expertos no tienen una notación para representar a las reglas?**

**R** Sí, y allí está el problema. No hay una notación que sea la estándar.

www.elsolucionario.net

# Taller

Las preguntas de este taller verificarán su conocimiento en la aplicación del UML a las GUI y a los sistemas expertos. Las respuestas a los cuestionarios podrán encontrarse en el Apéndice A, "Respuestas a los cuestionarios".

## Cuestionario

1. ¿Cuáles son las ventajas de nuestro modelo de una GUI?
2. ¿Cuáles son los componentes de un sistema experto?
3. ¿Qué características de un sistema experto comprende nuestro diagrama?

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# PARTE IV
# Apéndices

## Apéndice

A Respuestas a los cuestionarios

B Herramientas de modelado para el UML

C Un resumen gráfico

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# APÉNDICE A

# Respuestas a los cuestionarios

## Hora 1

1. ¿Por qué es necesario contar con diversos diagramas en el modelo de un sistema?

   Cualquier sistema cuenta con diversos usuarios con intereses distintos. Cada tipo de diagrama UML presenta una idea que podrá ser comprendida por cualquiera de los usuarios.

2. ¿Cuáles diagramas le dan una perspectiva estática de un sistema?

   Los diagramas de clases, objetos, componentes y distribución.

3. ¿Cuáles diagramas le dan una perspectiva dinámica de un sistema (esto es, muestran el cambio progresivo)?

   Los diagramas de casos de uso, estados, secuencias, actividades y colaboraciones.

www.elsolucionario.net

www.elsolucionario.net

# Hora 2

1. ¿Qué es un objeto?

   Un objeto es una instancia de una clase.

2. ¿Cómo trabajan los objetos en conjunto?

   Los objetos trabajan en conjunto mediante el envío de mensajes entre sí.

3. ¿Qué establece la multiplicidad?

   La multiplicidad establece la cantidad de objetos de una clase que se relacionan con otro de una clase asociada.

4. ¿Pueden asociarse dos objetos entre sí de más de una manera?

   Sí, por ejemplo: dos personas pueden estar asociadas como amigos y colaboradores.

# Hora 3

1. ¿Cómo representa una clase en el UML?

   Con un rectángulo; el nombre de la clase se coloca dentro de él, en la parte superior.

2. ¿Qué información puede mostrar en un símbolo de clase?

   Puede colocar los atributos, operaciones y responsabilidades de la clase.

3. ¿Qué es una restricción?

   Es una regla y se escribe entre llaves.

4. ¿Para qué adjuntaría una nota a un símbolo de clase?

   Para agregar información que no se encuentra en los atributos, operaciones o responsabilidades. Por ejemplo, podría desear que el usuario del modelo lea un documento en particular que contenga información respecto a la clase.

www.elsolucionario.net

# Hora 4

1. ¿Cómo representaría la multiplicidad?

   En uno de los extremos de la línea de asociación, coloque en el extremo lejano la cantidad de objetos que provienen de la clase que se relacionen con un objeto del extremo próximo.

2. ¿Cómo descubrirá la herencia?

   En la lista de clases de su modelo inicial, localice dos o más clases que compartan atributos y operaciones. Ya sea que otra clase de su modelo inicial se convierta en la clase principal de las clases que comparten atributos, o que tenga que crear una clase principal.

www.elsolucionario.net

3. ¿Qué es una clase abstracta?

   Una clase abstracta es aquella que funciona como la base de la herencia, aunque no provee objetos.

4. ¿Cuál es el efecto de un calificador?

   El efecto de un calificador es reducir una multiplicidad de uno a muchos a una de uno a uno.

# Hora 5

1. ¿Cuál es la diferencia entre una agregación y una composición?

   Tanto una agregación como una composición especifican una asociación de componentes que conforman a un todo. En una agregación, un componente puede ser parte de más de un todo. En una composición, un componente sólo puede ser parte de un todo.

2. ¿Qué es la realización?

   Es la relación entre una clase y una interfaz. Se dice que la clase realiza las operaciones en la interfaz.

3. Mencione los tres niveles de visibilidad y describa lo que significa cada uno de ellos.

   Si los atributos y operaciones de una clase tienen una visibilidad pública, pueden ser utilizados por otra. Si la visibilidad está protegida, una clase secundaria (u otra descendiente) podría utilizarlos. Si son privados, sólo la clase que los contiene podrá utilizarlos. Las operaciones de una interfaz tienen una visibilidad pública.

# Hora 6

1. ¿Cómo se llama a la entidad que inicia un caso de uso?

   Se denomina *actor*.

2. ¿Qué se entiende con "incluir un caso de uso"?

   Se entiende que alguno de los pasos en una situación dentro de un caso de uso son los mismos que los de otro y en lugar de listar los mismos pasos, tan sólo indicamos el caso de uso de donde provienen.

3. ¿Qué se entiende con "extender un caso de uso"?

   Se entiende que se agregan pasos a un caso de uso existente, y esto se hace para crear un caso de uso nuevo.

4. ¿Un caso de uso es lo mismo que un escenario?

   No. Un caso de uso es una colección de escenarios.

www.elsolucionario.net

www.elsolucionario.net

# Hora 7

1. Mencione dos ventajas de concebir un caso de uso.

   La primera ventaja de la visualización es que usted puede mostrar los casos de uso a los usuarios y lograr que le den información adicional, y la segunda ventaja es que puede combinar los diagramas de casos de uso con otro tipo de diagramas.

2. Describa a la generalización y el agrupamiento, relaciones entre los casos de uso que ha visto durante esta hora. Mencione dos situaciones en las que usted agruparía los casos de uso.

   En la generalización, un caso de uso hereda el significado y comportamientos de otro. El agrupamiento es la organización de un conjunto de casos de uso dentro de paquetes.

3. ¿Cuáles son las similitudes entre las clases y los casos de uso? ¿Cuáles las diferencias?

   Similitudes: ambos son elementos estructurales. Ambos pueden heredar. Diferencias: La clase consta de atributos y operaciones. El caso de uso consta de escenarios y cada uno consta de una secuencia de pasos. La clase proporciona una idea estática de las partes del sistema, el caso de uso una idea dinámica. La clase muestra el interior del sistema. El caso de uso muestra el aspecto del sistema a alguna persona.

# Hora 8

1. ¿De qué forma difiere un diagrama de estados de uno de clases, de objetos o de casos de uso?

   Un diagrama de estados modela los estados de un solo objeto. Un diagrama de clases, de objetos o de caso de uso modela un sistema, o al menos parte de él.

2. Defina los siguientes términos: *transición*, *suceso* y *acción*.

   Una transición es un cambio de un estado a otro. Un evento es un suceso que provoca una transición. Una acción es un proceso ejecutable que resulta de un cambio de estado.

3. ¿Qué es una *transición no desencadenada*?

   Es una transición que ocurre por las actividades dentro de un estado, en lugar de ocurrir como respuesta a un evento.

4. ¿Cuál es la diferencia entre los subestados secuenciales y los concurrentes?

   Los subestados son estados dentro de otros. Los subestados secuenciales suceden uno después de otro. Los subestados concurrentes suceden al mismo tiempo.

www.elsolucionario.net

www.elsolucionario.net

# Hora 9

1. Defina mensaje sincrónico y mensaje asincrónico.

   Cuando un objeto envía un mensaje sincrónico, aguarda una respuesta antes de continuar. En el caso del mensaje asincrónico, el objeto no aguarda una respuesta.

2. En un diagrama de secuencias genérico ¿cómo representaría el control de flujo implícito en una instrucción condicional?

   Se coloca la condición entre corchetes.

3. ¿Cómo representaría el control de flujo implícito en una instrucción de ciclo "mientras"?

   Se coloca la condición entre corchetes y se antecede al corchete izquierdo con un asterisco.

4. En un diagrama de secuencias ¿Cómo representaría a un objeto recién creado?

   Se representa por un rectángulo de objeto colocado en el tiempo de actividad (esto es, en algún lugar de la dimensión vertical) de modo que su ubicación represente el momento en que se haya creado en la secuencia. Agregue "Crear()" o «Crear» a la flecha del mensaje que apunte al objeto generado.

# Hora 10

1. ¿Cómo representa un mensaje en un diagrama de colaboraciones?

   Por una flecha junto a la línea de asociación que une a un par de objetos. La flecha apunta al objeto receptor.

2. ¿Cómo mostraría información secuencial en un diagrama de colaboraciones?

   Adjuntando un número al rótulo de una flecha de mensaje. El número corresponde al orden secuencial del mensaje.

3. ¿Cómo mostraría los cambios de estado?

   Dentro del rectángulo de un objeto, indique su estado. Agregue otro rectángulo al objeto y muestre el estado modificado. Conecte a ambos con una línea punteada y rotule la línea con un estereotipo «se convierte en».

4. ¿Qué se entiende por la "equivalencia semántica" de dos tipos de diagramas?

   Ambos tipos de diagramas muestran la misma información y podrá convertir uno en otro.

www.elsolucionario.net

www.elsolucionario.net

# Hora 11

1. ¿Cuáles son las dos formas de representar un punto de decisión?

   Una de ellas es mostrar un rombo con bifurcaciones provenientes de él. La otra es mostrar las bifurcaciones provenientes directamente de una actividad. De cualquier forma, coloque una condición entre corchetes en cada ramificación.

2. ¿Qué es un marco de responsabilidad?

   En un diagrama de actividad, es un segmento que muestra las actividades que realiza algún rol en particular.

3. ¿Cómo representaría la transmisión y recepción de una indicación?

   Utilice un pentágono convexo para mostrar la transmisión de una indicación, y uno cóncavo para representar la recepción.

# Hora 12

1. ¿Cuáles son los tres tipos de componentes?

   Son: componentes de distribución, de producto de trabajo y de ejecución.

2. ¿Cómo llamaría a la relación entre un componente y su interfaz?

   Se conoce como *realización*.

3. ¿Cuáles son las dos formas de representar esta relación?

   La primera muestra la interfaz como un rectángulo que contiene la información que se le relaciona, se conecta al componente por la línea discontinua y una punta de flecha representada por un triángulo sin rellenar que visualiza la realización. La segunda representará la interfaz como un pequeño círculo que se conecta al componente por una línea continua. En este contexto, la línea representa la relación de realización.

4. ¿Qué es una *interfaz de exportación*? ¿Que es una *interfaz de importación*?

   La interfaz de exportación es una interfaz que un componente pone a disposición de otros componentes de modo que puedan utilizar sus servicios. Cuando otro componente utiliza tales servicios, se convertirá en una interfaz de importación.

# Hora 13

1. ¿Cómo representa a un nodo en un diagrama de distribución?

   Con un cubo.

www.elsolucionario.net

www.elsolucionario.net

2. ¿Qué tipo de información puede aparecer en un nodo?

   Puede estar el nombre del nodo, del paquete, y los componentes distribuidos en el nodo.

3. ¿Cuáles son los dos tipos de nodos?

   Los procesadores (que pueden ejecutar componentes) y dispositivos (que se conectan con el mundo exterior).

4. ¿De qué forma funciona una red token-ring?

   Esta red conecta a las computadoras equipadas con una tarjeta de red a una unidad central de accesso a multiestaciones a unidades de acceso a varias estaciones (MSAU) conectadas a manera de anillo. Las MSAU pasan una señal conocida como Token por el anillo. La posición de la señal indica el equipo que puede enviar información en algún momento.

# Hora 14

1. ¿Cuáles son las cuatro capas del UML?

   Son: la capa de objetos del usuario, la de modelo, la de metamodelo, y la de metametamodelo.

2. ¿Qué es un clasificador?

   Es cualquier elemento que defina una estructura y comportamiento.

3. ¿Por qué es importante poder extender al UML?

   Cuando empiece a utilizar el UML para modelar sistemas reales, se encontrará con situaciones que son más amplias y complejas que las que verá en las referencias y libros de textos. Si puede extender el UML, podrá reflejar la naturaleza de tales situaciones reales.

4. ¿Cuáles son los mecanismos de extensión del UML?

   Son los estereotipos, las restricciones y los valores rotulados o etiquetados.

# Hora 15

1. ¿Cuáles son algunas de las inquietudes de un cliente?

   ¿El equipo de desarrollo comprendió el problema? ¿Acaso los miembros del equipo comprenden la idea del cliente para resolverlo? ¿Qué productos del trabajo puede esperar el cliente del equipo de desarrollo? ¿De qué forma el administrador del proyecto informa al cliente? ¿Qué tanto puede profundizar el equipo en cualquier punto?

www.elsolucionario.net

www.elsolucionario.net

2. ¿Qué se debe comprender como *metodología* de desarrollo?

   Una metodología de desarrollo establece la estructura y naturaleza de los pasos en un proyecto de desarrollo.

3. ¿Cuál es el método "en cascada"? ¿Cuáles son sus debilidades?

   En este método, el análisis, diseño, codificación y distribución son pasos secuenciales y no recurrentes.

4. ¿Cuáles son los segmentos de GRAPPLE?

   Son: Recopilación de necesidades, Análisis, Diseño, Desarrollo y Distribución.

5. ¿Qué es una sesión JAD?

   Una sesión JAD (Desarrollo conjunto de aplicaciones) reúne a quienes toman decisiones dentro de la organización del cliente, a usuarios potenciales del sistema, y a los miembros del equipo de desarrollo. Algunas sesiones JAD conjugan al equipo de desarrollo con sólo los usuarios.

# Hora 16

1. ¿Cuál diagrama del UML es adecuado para modelar un proceso del negocio?

   El diagrama de actividades.

2. ¿Cómo podría modificar este diagrama para mostrar qué hace cada quién?

   Puede utilizar el diagrama de actividades para agregarle marcos de responsabilidades. Cada responsable se coloca en la parte superior del marco que le corresponda.

3. ¿Qué debe entenderse por "lógica de negocios"?

   Es un conjunto de normas que siguen los negocios en situaciones específicas.

# Hora 17

1. ¿De qué forma utilizaremos los sustantivos obtenidos en la entrevista con un experto?

   Los sustantivos se convierten en candidatos para nombres de clases y de atributos.

2. ¿Y de qué forma utilizaremos los verbos y construcciones verbales?

   Los verbos y construcciones verbales se convierten en candidatos para operaciones y para asociaciones de nombres.

3. ¿Qué es una asociación "tripartita"?

   Una asociación tripartita involucra a tres clases.

www.elsolucionario.net

www.elsolucionario.net

4. ¿Cómo modelaría una asociación tripartita?

   Mediante la vinculación de cada una de las clases en un rombo. Deberá escribir el nombre de la asociación cerca del rombo. Las multiplicidades en una asociación tripartita reflejan la cantidad de instancias de cualquier par de clases asociadas con una cantidad constante de instancias de la tercera.

# Hora 18

1. ¿Cómo hemos representado a las necesidades del sistema?

   Hemos usado el diagrama de paquetes junto con los casos de uso para lograrlo.

2. ¿Una vez que se hace el análisis del dominio ya finaliza el modelado de clases?

   El modelado de clases continúa su desarrollo luego del análisis del dominio.

3. ¿Qué es el "tiempo muerto"?

   Es el nombre que le he dado al tiempo que se pasa el mesero caminando por el restaurante. Sólo quise ver si usted ponía atención :-).

# Hora 19

1. ¿Cuáles son las partes de un diagrama de casos de uso típico?

   Son: el actor que inicia, el caso de uso, y el actor que se beneficia.

2. ¿A qué se refiere que un caso de uso "incluya" (o "utilice") a otro?

   "Incluir" un caso de uso significa que otro caso de uso incluye los pasos ya indicados en un caso de uso.

# Hora 20

1. ¿Cómo representaría a un objeto que se genera durante el curso de un diagrama de secuencias?

   Este objeto se representa colocándolo bajo el nivel de los demás objetos. Mejorará la claridad si agrega un estereotipo «crear» al mensaje que antecede al objeto.

2. ¿Cómo se representa al tiempo en un diagrama de secuencias?

   El tiempo se representa con la línea de vida del objeto.

www.elsolucionario.net

www.elsolucionario.net

3. ¿Qué es "la línea de vida"?

   Es una línea punteada que desciende de un objeto. Representa la existencia de un objeto en un cierto periodo.

4. En un diagrama de secuencias, ¿cómo muestra una "activación" y qué es lo que representa?

   Una activación se representa como un pequeño rectángulo en el tiempo de actividad de un objeto. Representa el periodo durante el cual el objeto realiza alguna acción.

# Hora 21

1. ¿Qué es un análisis de tareas?

   Es un estudio que realiza un diseñador de GUI para comprender lo que el usuario hará con la aplicación que corresponda con la interfaz.

2. ¿Cuál análisis de los que ya hemos hecho es un vago equivalente de un análisis de tareas?

   El análisis del caso de uso es equivalente al análisis de tareas.

3. ¿Qué se entiende por un diseño de tipo "pantalón de payaso"?

   Es un diseño de GUI que incorpora una cantidad excesiva de colores, tamaños de componentes y fuentes.

4. Dé tres razones para restringir el uso del color en una GUI.

   La primera razón es que la asociación de un color con un significado podría no ser tan obvio para el usuario, como lo es para el diseñador; en segundo lugar, demasiados colores distraerían a los usuarios de la tarea por realizar; y finalmente, parte de la población usuaria podría sufrir de daltonismo.

www.elsolucionario.net

# Hora 22

1. ¿Cómo representaría una clase parametrizada?

   Se utiliza la representación normal de la clase con un pequeño recuadro sobrepuesto en la esquina superior derecha. El pequeño recuadro está formado de líneas punteadas.

2. ¿Qué es "vincular" y cuáles son los dos tipos de vinculación?

   La "vinculación" es la acción de adjuntar un valor a un parámetro. Los dos tipos de vinculación son "explícita" e "implícita".

3. ¿Qué es un "patrón de diseño"?

   Un patrón de diseño es una solución probada a un problema de diseño. Se puede utilizar en diversas situaciones, y en el UML lo representa como una colaboración parametrizada.

www.elsolucionario.net

4. ¿Qué es el patrón de diseño "Cadena de responsabilidades"?

   En este patrón de diseño, un objeto cliente inicia una petición y la pasa al primero en una cadena de objetos. Si el primero de ellos no puede responder a la petición, la pasa al siguiente. Si tampoco la puede responder, la pasará al siguiente, y así hasta que un objeto pueda responder a la petición.

# Hora 23

1. ¿Qué es un sistema incrustado?

   Es un sistema de cómputo que se encuentra dentro de algún otro tipo de dispositivo, como un electrodoméstico.

2. ¿Qué es un evento asincrónico?

   Un evento es asincrónico cuando no se puede predecir en qué momento sucederá.

3. En términos de sistemas incrustado, ¿qué es un sistema "estricto" y qué es un sistema "tolerante"?

   Un sistema estricto debe cumplir con tiempos específicos, en tanto que el tolerante no.

4. ¿Qué sucede en un "núcleo preferencial"?

   En este tipo de núcleo, luego que se ejecuta una ISR, la CPU no regresa al subproceso interrumpido si hay un subproceso de mayor prioridad en estado de Listo. En vez de ello, ejecutará al subproceso de mayor prioridad.

# Hora 24

1. ¿Cuáles son las ventajas de nuestro modelo de una GUI?

   Nuestro modelo puede capturar nuestros procesos conceptuales en la evolución de una GUI y centra la atención en los casos de uso conectados con cada pantalla.

2. ¿Cuáles son los componentes de un sistema experto?

   Son: base de conocimientos, área de trabajo y motor de interfaz.

3. ¿Qué características de un sistema experto comprende nuestro diagrama?

   Nuestro diagrama muestra las partes de una regla, las reglas asociadas y las relaciones entre reglas.

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# APÉNDICE B

# Herramientas de modelado para el UML

Conforme leyó el tema del libro y realizó los ejercicios, posiblemente utilizó un lápiz y papel para crear sus diagramas. Si necesitara hacerlo en sus procesos de modelado relacionados con proyectos, pronto se encontraría con problemas. Además de tener que dibujar todas esas líneas, círculos, elipses y rectángulos, tendrá dificultades cuando desee modificar la disposición de un diagrama ya terminado.

Afortunadamente, la tecnología viene al rescate. Existen diversas herramientas que le pueden ayudar a crear modelos UML. Este apéndice le hablará de tres de ellas: Rational Rose, SELECT Enterprise y Visual UML. No veremos todas sus características y opciones, sin embargo veremos un poco de cada una para que tenga una idea general de la forma en que funcionan y lo que hacen.

## Características en común

Aunque estas herramientas tienen características muy diversas, cuentan con algunas características en común. Cada una permite una diagramación de

www.elsolucionario.net

www.elsolucionario.net

tipo "banda elástica": al crear un vínculo entre dos elementos, éste se ajustará cuando los arrastre por la pantalla. Cada uno permite al menos cierta generación de código, con lo que se producen pequeñas secciones de código a partir de los modelos. Finalmente, todos ellos cuentan con extensas herramientas y cuadros de diálogo para las modificaciones.

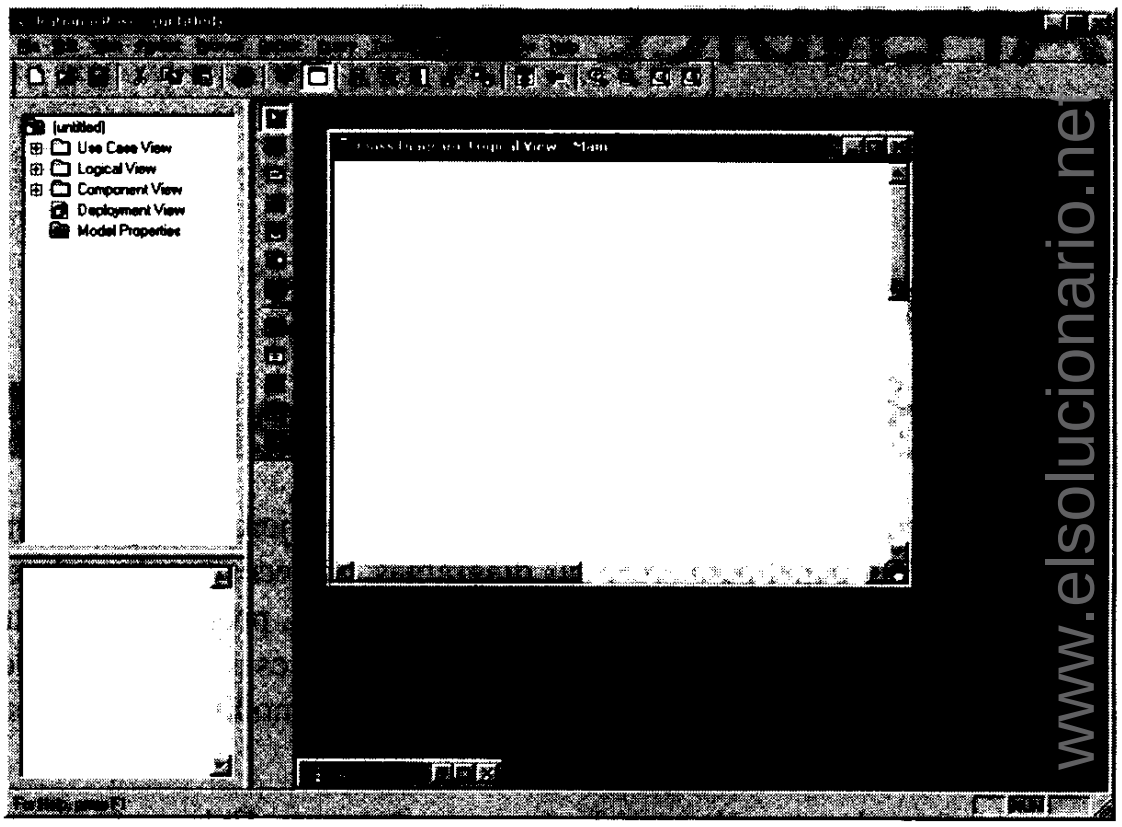
# Rational Rose

El líder del mercado en las herramientas de modelado, Rational Rose, es un producto de la empresa que da empleo a los Tres Amigos.

Cuando ejecute el programa, verá la ventana de la figura B.1. Como podrá ver, las demás herramientas tendrán ventanas similares.

**FIGURA B.1**

*Pantalla inicial de Rational Rose.*



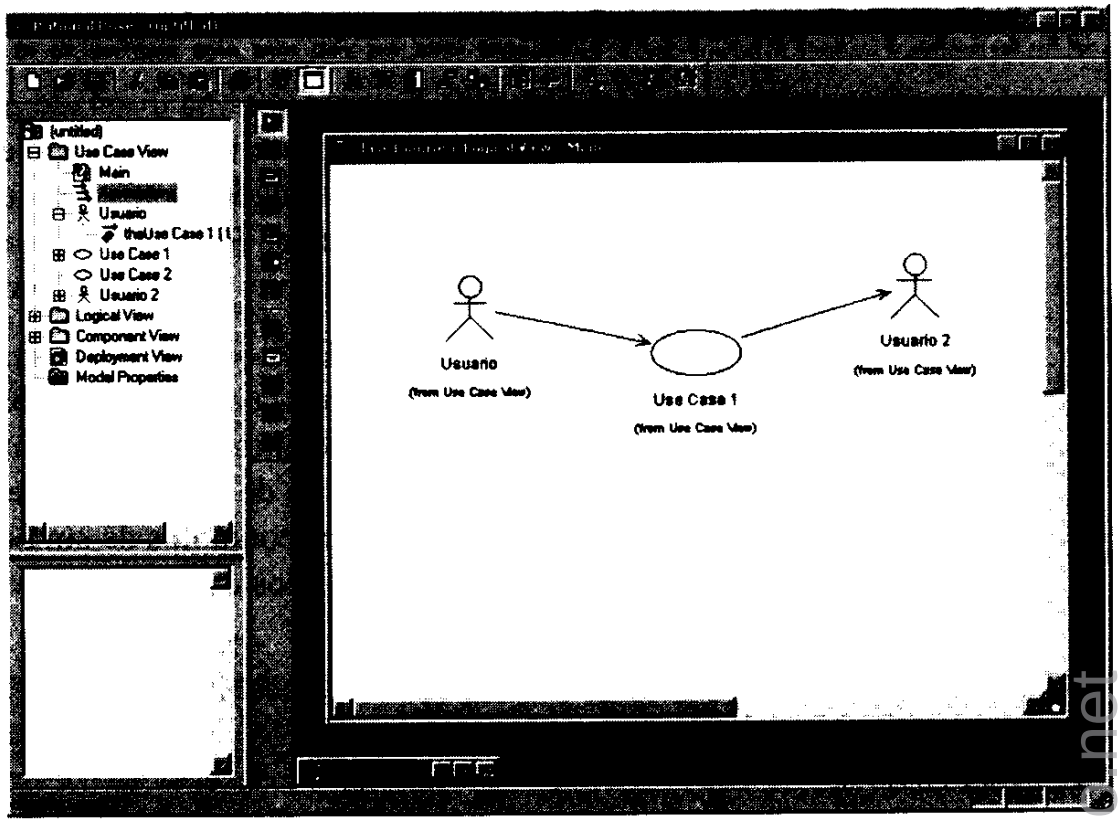

El panel de la izquierda es el examinador y el de la derecha contiene la ventana de diagramación; entre ellos se encuentra una paleta de iconos UML para un diagrama de clases, el cual es el tipo de diagrama que se abre de forma predeterminada.

El examinador está organizado de acuerdo con vistas: vista de Caso de uso, vista Lógica, vista de Componentes, y vista de Distribución. Para crear un diagrama, deberá hacer clic con el botón derecho en las vistas, y las opciones de diagramación relacionadas con ellas aparecerán en un menú. Seleccione una y aparecerá una nueva ventana de diagramación. También contará con una paleta de iconos adecuada para el diagrama que utilice. La figura B.2 muestra un diagrama de casos de uso y su paleta.

www.elsolucionario.net

**FIGURA B.2**

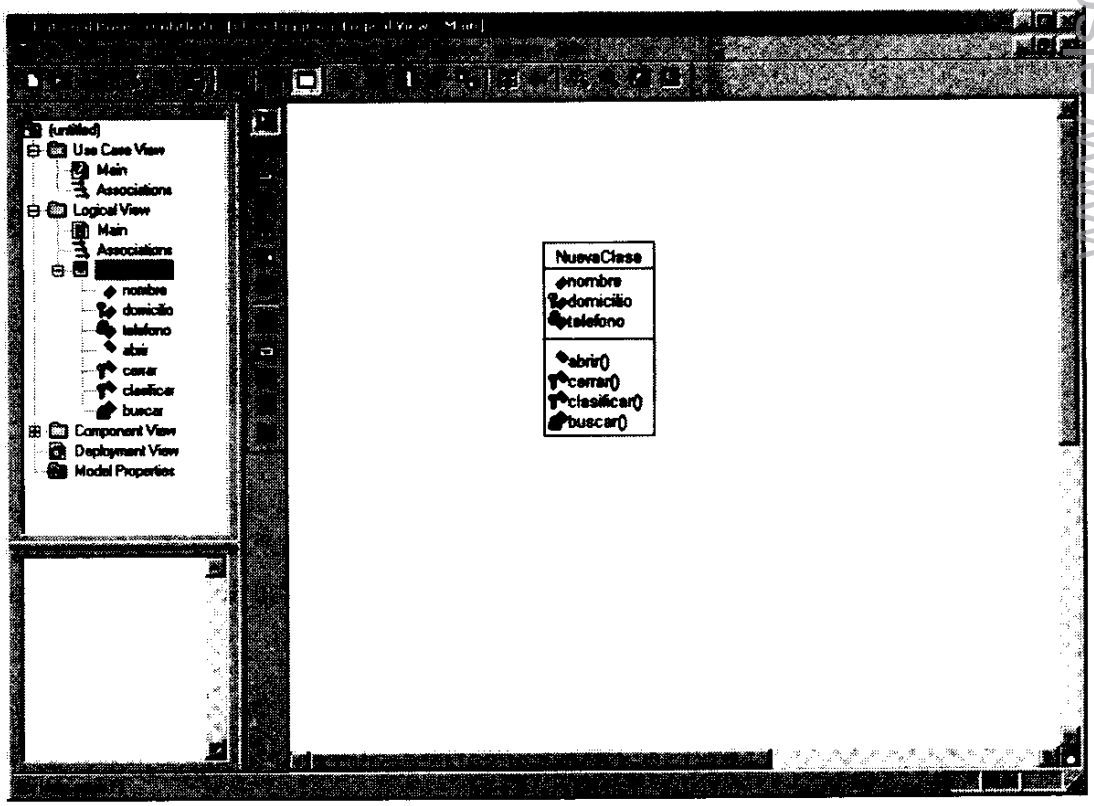
*Un diagrama de casos de uso y su paleta en Rose.*




Rose agrega cierto dinamismo a sus iconos. Cuando desee mostrar los atributos y las operaciones de una clase como públicos, privados o protegidos, podrá agregarles distintivos visuales como los que aparecen en la figura B.3.

**FIGURA B.3**

*Rose establece algunos distintivos visuales para los iconos de clases.*




Para obtener una mayor información de Rational Rose, visite `http://www.rational.com`.

www.elsolucionario.net

www.elsolucionario.net

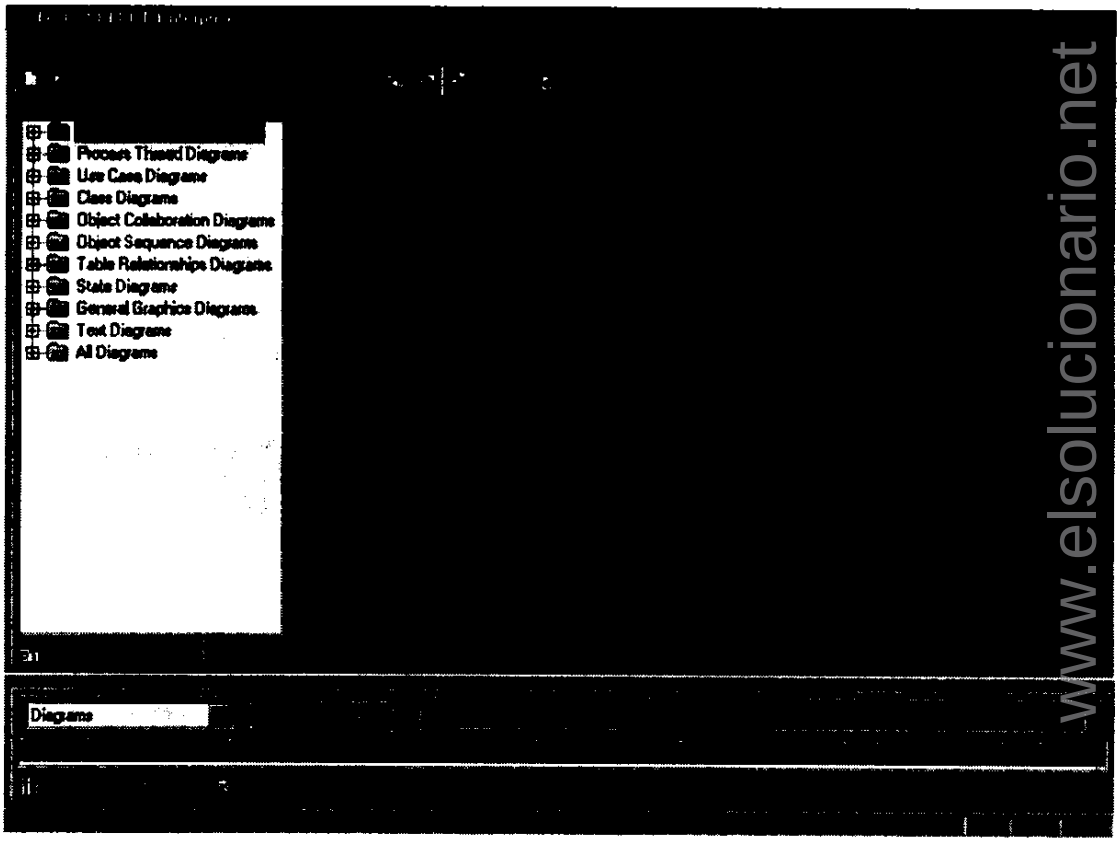
# SELECT Enterprise

Como su nombre lo indica, SELECT Enterprise se ha diseñado para ser una herramienta de modelado para toda una empresa. En ese sentido, automáticamente se conecta con un depósito en red donde es accesible para los modeladores y los usuarios de los modelos de toda una empresa. Las ventajas de utilizar un depósito son que se lleva un control de las personas que acceden a los modelos y permite llevar un control de versiones.

Cuando abra un modelo, la ventana de SELECT (vea la figura B.4) cargará un entorno parecido al de Rose, pero sólo hasta allí queda la similitud. Ambas herramientas tienen diferentes perspectivas: mientras que Rose se enfoca estrictamente en el UML, en el esquema de cosas de SELECT, el UML es uno de varios paquetes de modelado importantes. SELECT Enterprise le permite utilizar conjuntos de símbolos que no son de UML para dibujar modelos de datos y generar modelos de procesos de negocios.

**FIGURA B.4**

*La pantalla de SELECT Enterprise cuando se abre un modelo.*



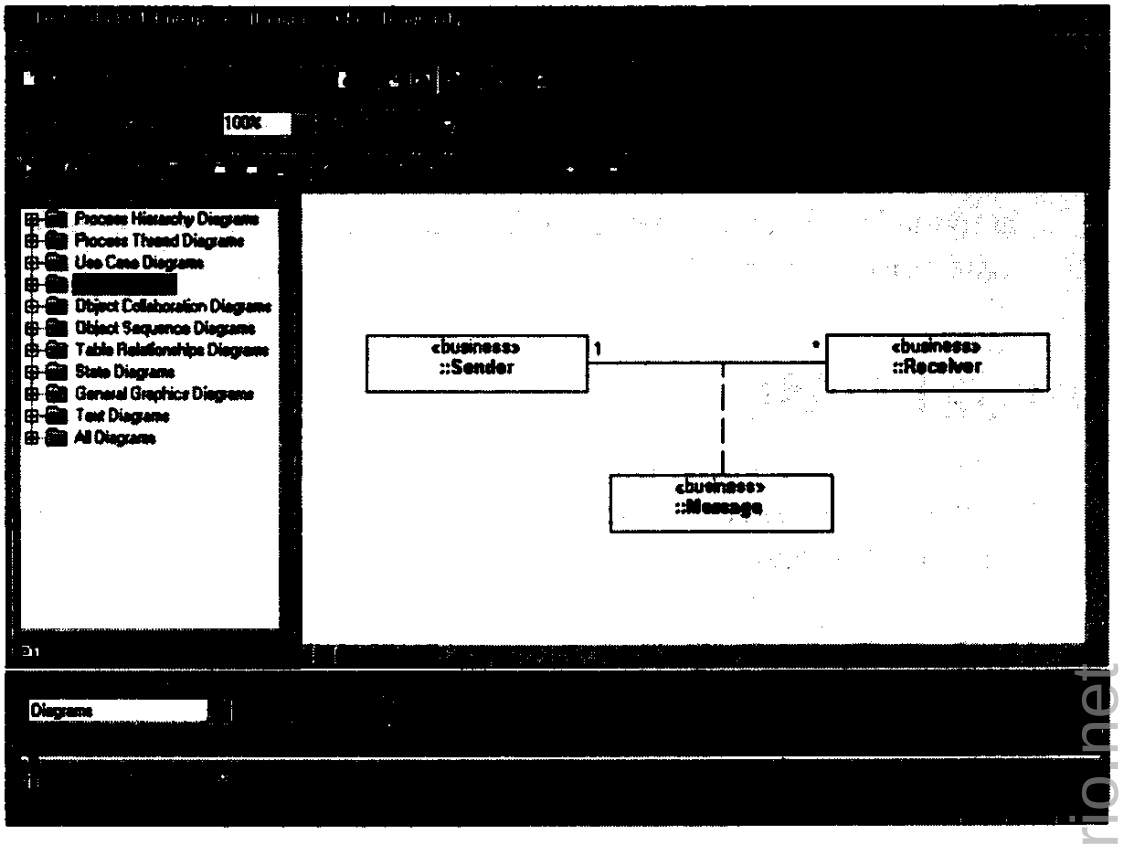

La ventana de exploración de SELECT es el equivalente al examinador de Rose. Consta de cuatro fichas. Una de ellas contiene el diccionario de su modelo, la otra le permite crear sus diagramas de modelado. La tercer ficha le ayuda a organizar sus diagramas, y cuarta que le otorga ayuda.

La figura B.5 le muestra un diagrama de clases en SELECT. Observe los estereotipos de «business»: aparecen de forma predeterminada cuando usted genera una clase.

www.elsolucionario.net

**FIGURA B.5**

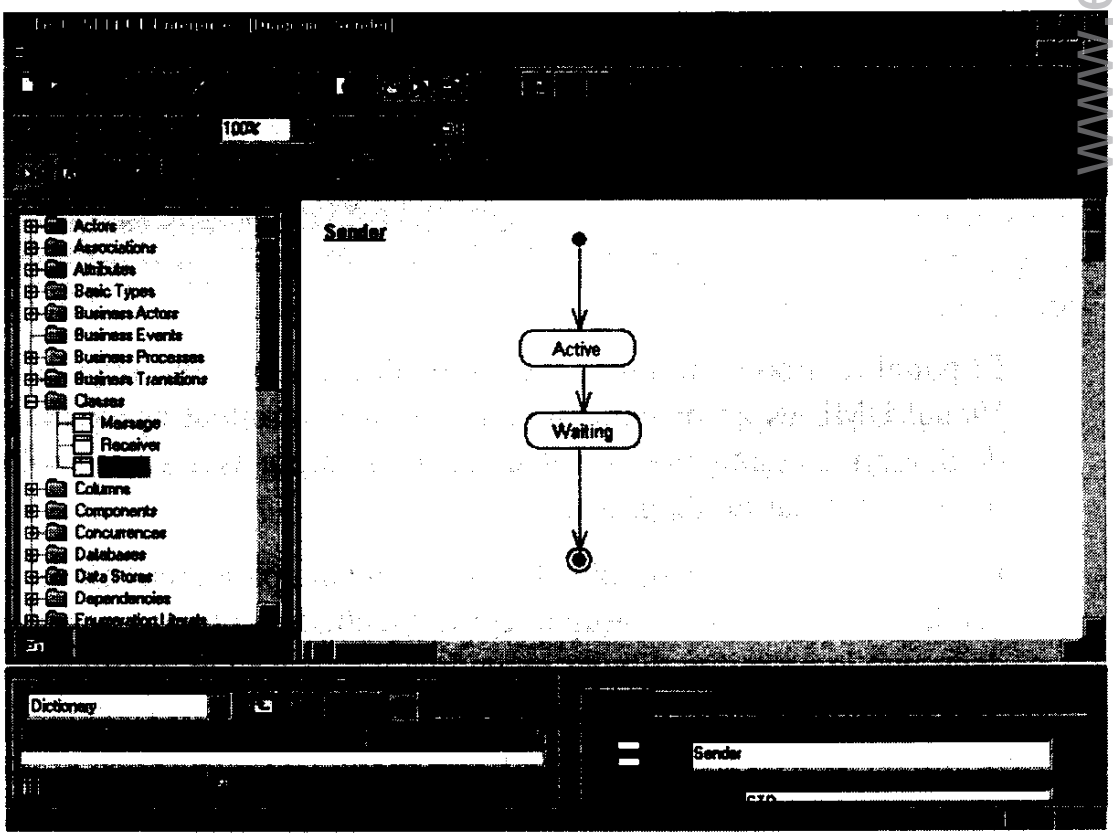
*Un diagrama de clases en SELECT.*




Aparte de su propio funcionamiento intrínseco, un diagrama de clases de SELECT es importante por otra razón: SELECT le permite crear un diagrama de estados sólo si ya ha creado una clase por adjuntarle. La figura B.6 le muestra un diagrama de estado. Observe el nombre de la clase del diagrama de estados en la esquina superior izquierda del diagrama.

**FIGURA B.6**

*Un diagrama de estados en SELECT.*




www.elsolucionario.net

Para enriquecer su modelo, puede vincular elementos de modelado entre sí. También podrá vincular diagramas y podrá ejecutar un informe que muestre cuáles diagramas están vinculados a uno en particular.

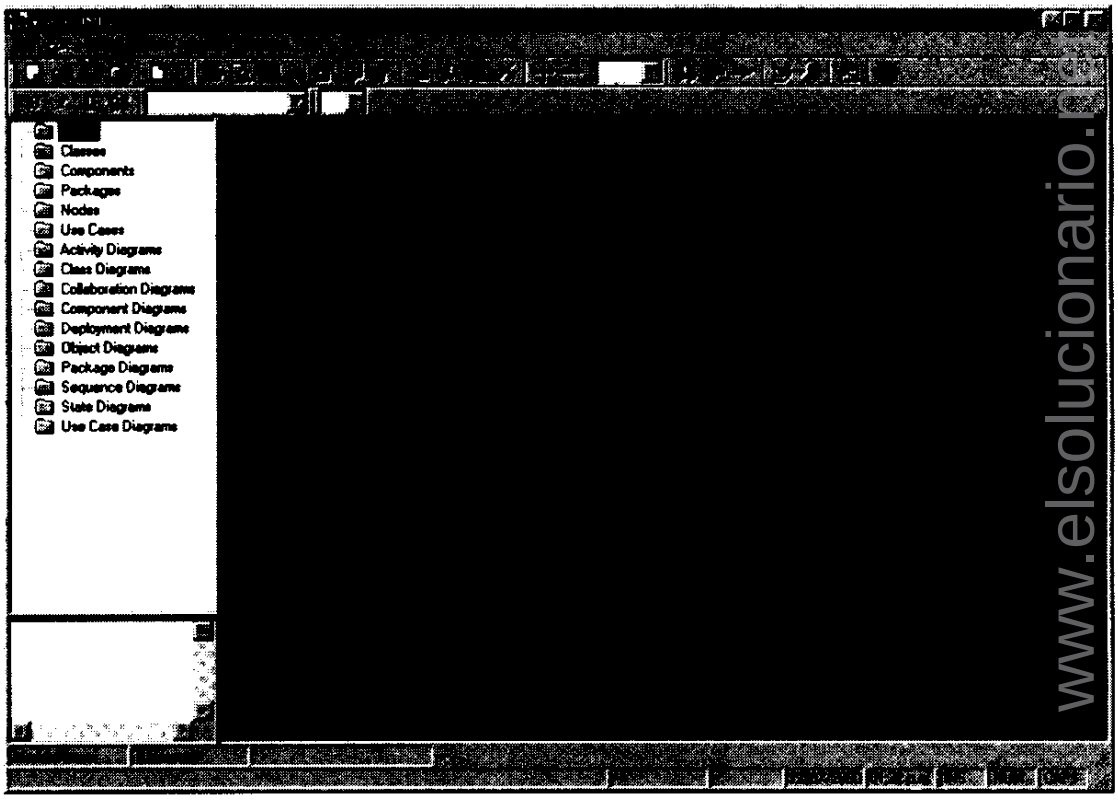
Cuando escribí estas líneas, SELECT Enterprise no permitía el uso de diagramas de actividades, de componentes o de distribución. Para mayor información de SELECT Enterprise, diríjase al sitio `http://www.selectst.com`.

# Visual UML

Cuando ejecute Visual UML (vea la figura B.7), verá una interfaz ordenada, la cual le aparecerá con toda simplicidad. Si conoce algo de UML, estará dibujando diagramas en Visual UML en muy poco tiempo.

**FIGURA B.7**

*Pantalla inicial de Visual UML.*



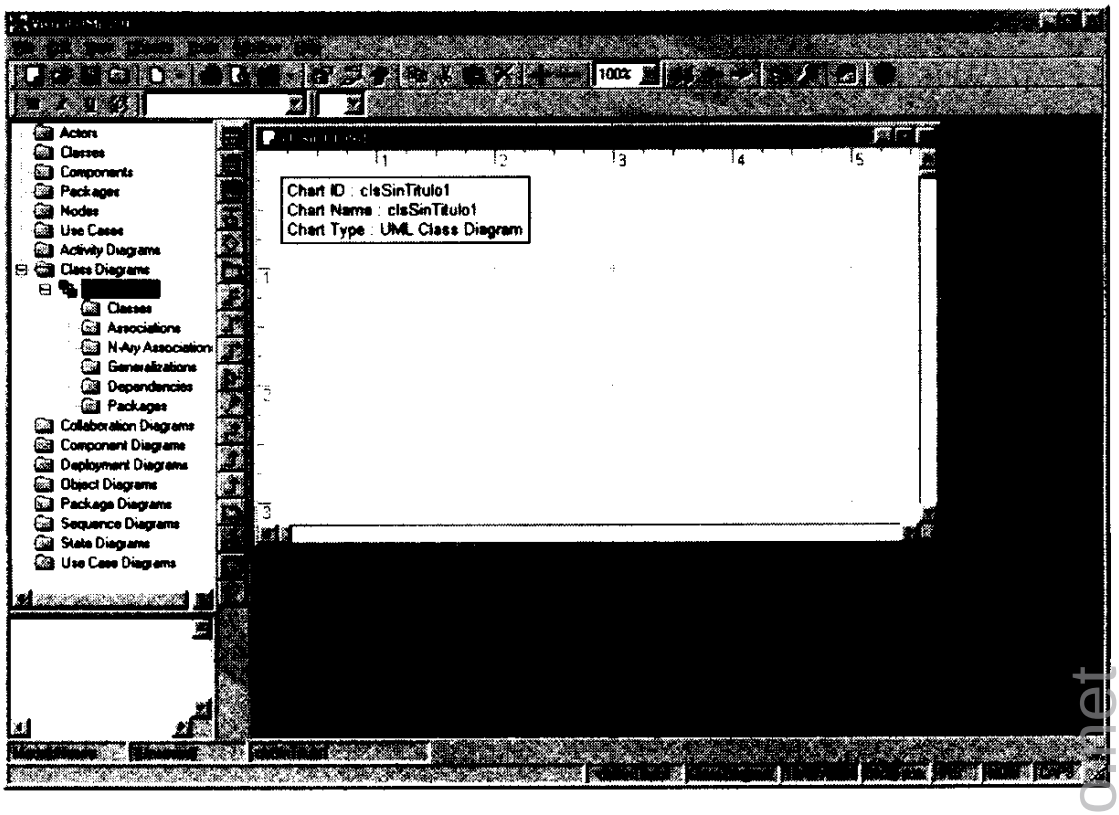

El panel de tipo explorador y examinador del extremo izquierdo de la ventana del Visual UML es un mecanismo para crear y administrar los diagramas. Todos los tipos de diagramas están en el nivel superior. Siempre estará a tan sólo un par de clics del ratón para crear un diagrama.

La figura B.8 le muestra la paleta y la ventana para crear un diagrama de clase. Las paletas de esta herramienta otorgan cierta facultad de utilizar imágenes que no son del UML, dado que le permiten dibujar recuadros y líneas en el diagrama.

www.elsolucionario.net

**FIGURA B.8**

*Creación de un diagrama de clases en Visual UML.*




Cabe hacer notar que podrá vincular los objetos de un diagrama con los de otro.

Visite `http://www.visualuml.com` para conocer mayores detalles de Visual UML.

# La herramienta ideal para el modelado

Cada una de estas herramientas de modelado cuenta con una buenas características; pero aún así, pienso que se les podrían agregar más.

Una de ellas podría ser flexibilidad. Con frecuencia quisiera utilizar los iconos de un tipo de diagrama en otro para así conformar un diagrama híbrido. Los iconos de estado podrían ser muy útiles en un diagrama de secuencias. La representación de un Actor en un diagrama de casos de uso podría ser muy útil en otros diagramas. Aunque ésta podría ser una facultad muy difícil de incorporar, pagaría enormes dividendos entre los modeladores.

Otra característica que me gustaría ver es... mayor flexibilidad. Además de importar iconos de un tipo de diagrama a otro, me gustaría poder importar imágenes prediseñadas y utilizarlas como estereotipos gráficos en los diagramas. Quizá algunas imágenes utilizadas con frecuencia podrían incluirse en la herramienta.

Finalmente, un manual de instrucción metódico y con cierta animación podría ser de mucha utilidad en cualquier herramienta de modelado. El manual de instrucción podría presentar ideas elementales respecto al UML y mostrar cómo se orientan dentro de las facultades propias de la herramienta de modelado. Esto podría convertirse en un excelente servicio para las personas que recién ingresan al mundo del modelado.

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

www.elsolucionario.net

# APÉNDICE C

# Un resumen gráfico

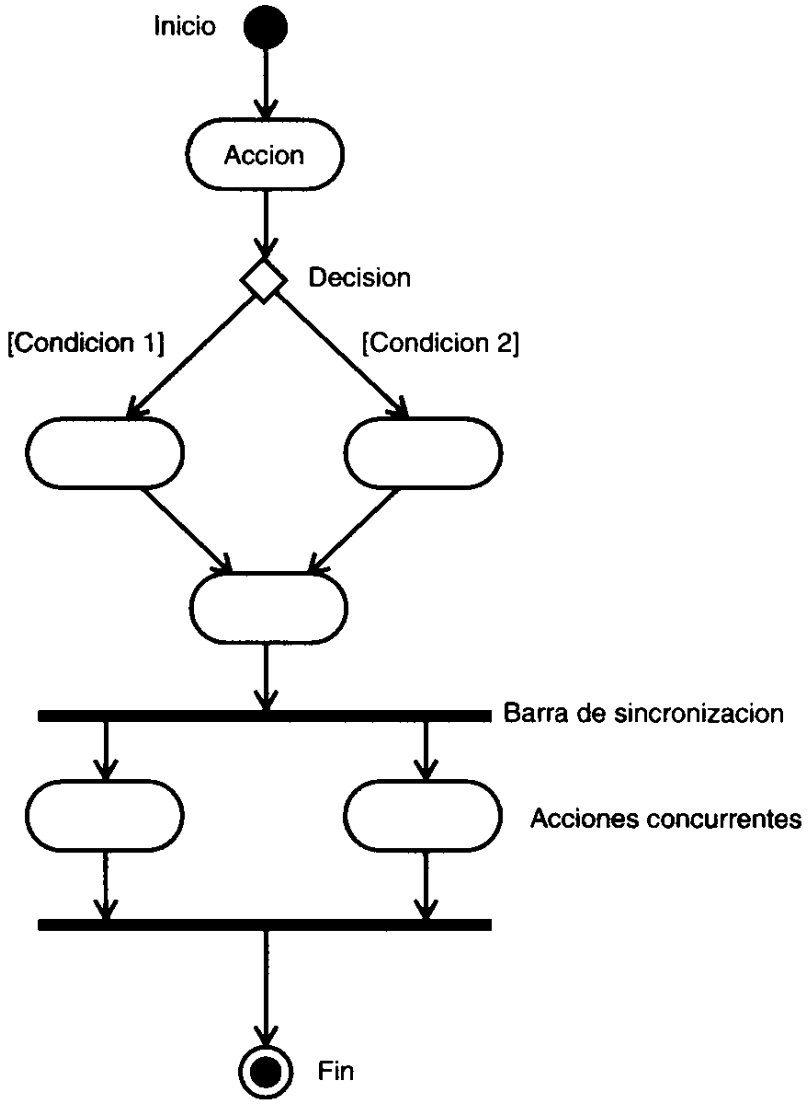
Este apéndice presenta algunos de los aspectos primordiales de cada diagrama UML.

www.elsolucionario.net

www.elsolucionario.net

# Diagrama de actividades

**FIGURA C.1**




www.elsolucionario.net

www.elsolucionario.net

**FIGURA C.2**

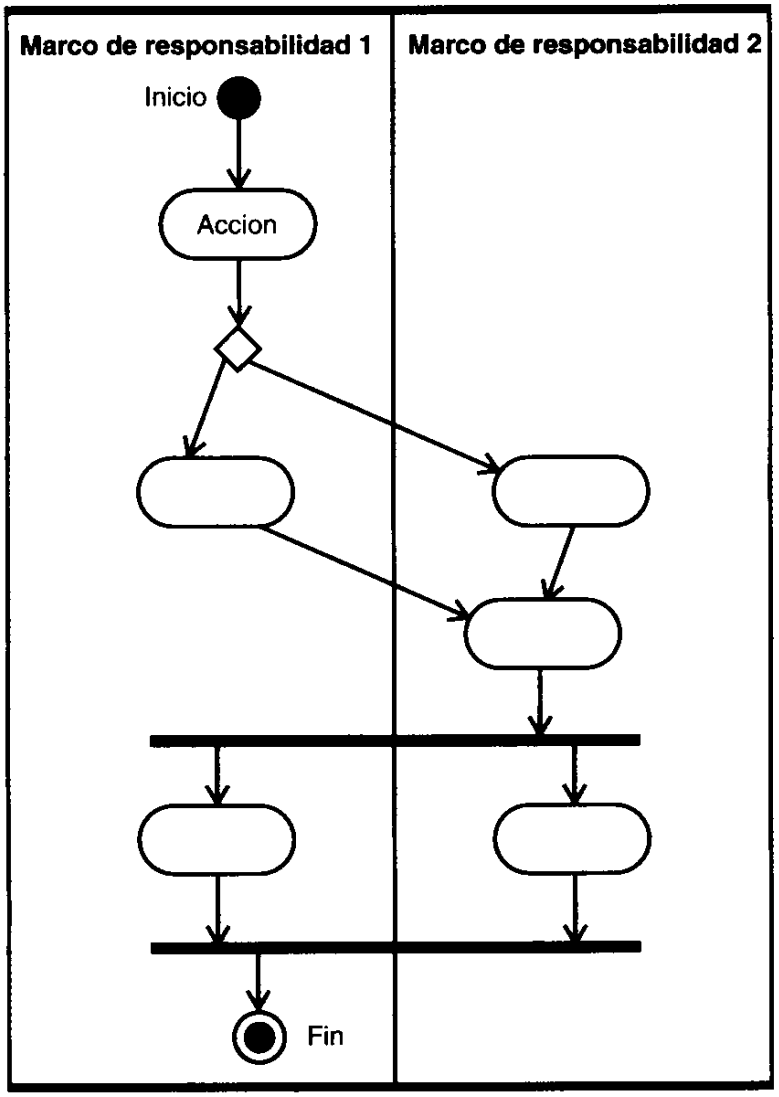
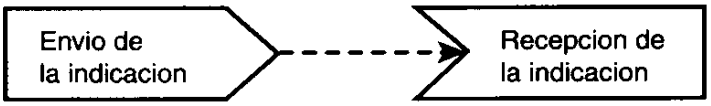



**FIGURA C.3**

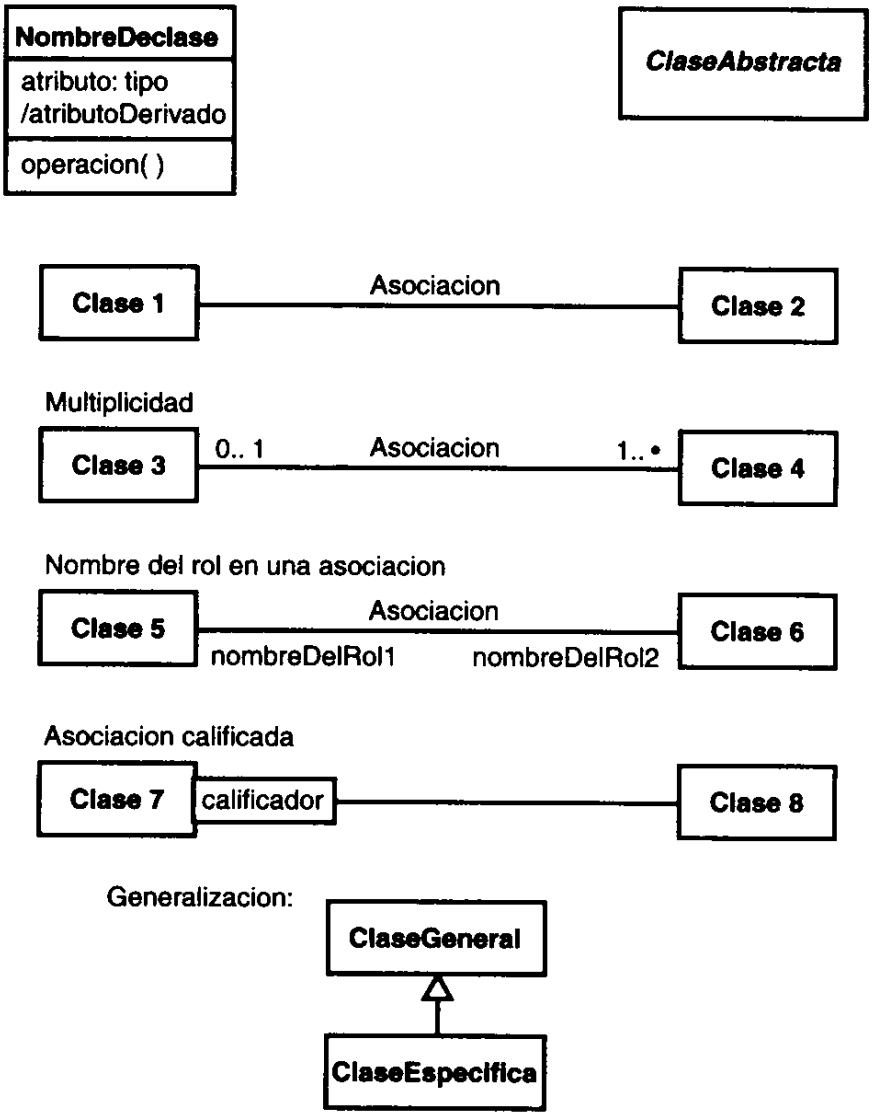



www.elsolucionario.net

www.elsolucionario.net

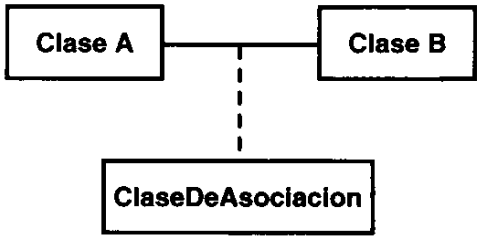
# Diagrama de clases

**FIGURA C.4**

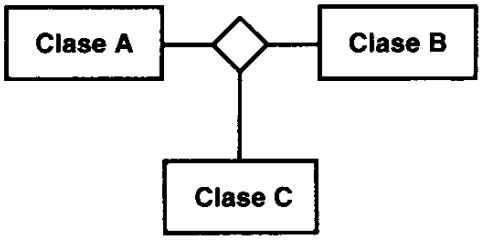



www.elsolucionario.net

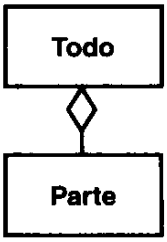
**FIGURA C.5**



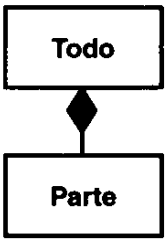

Asociacion tripartita:

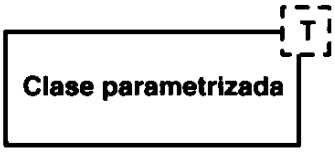



Agregacion:




Composicion:

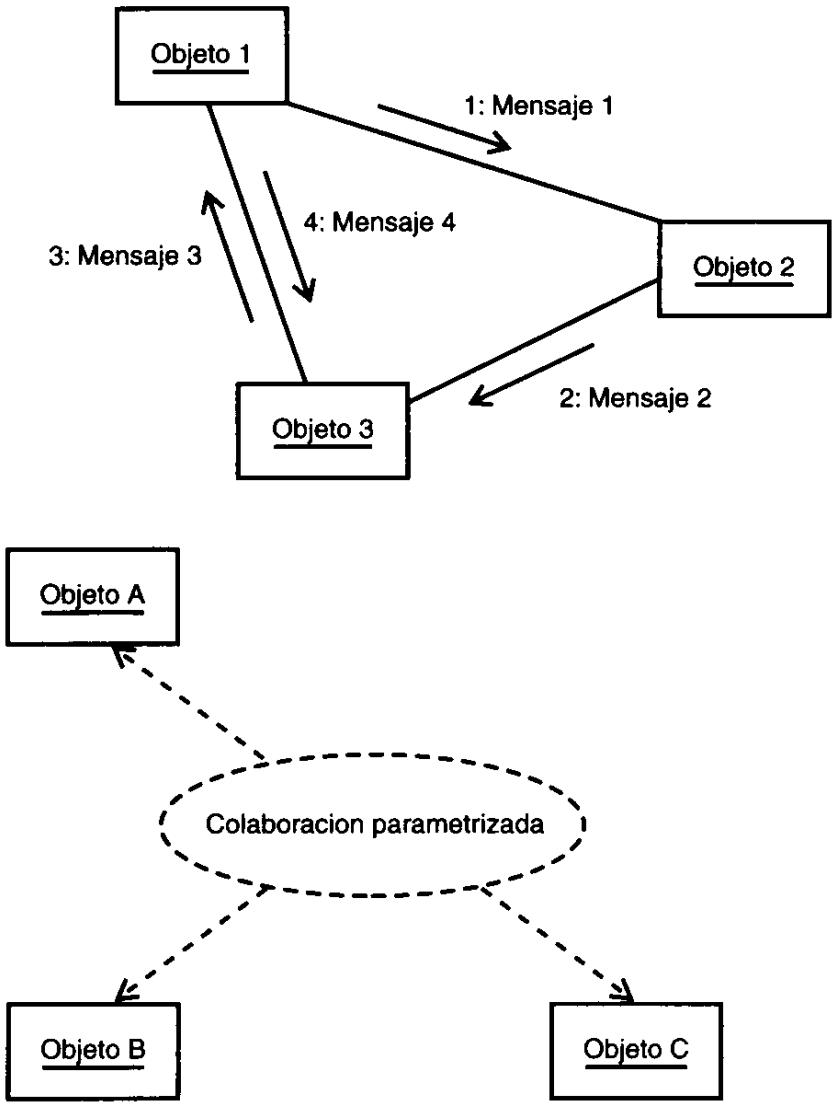



www.elsolucionario.net

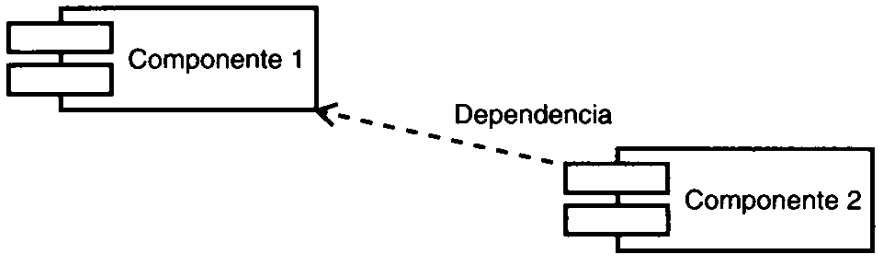
# Diagrama de colaboraciones

**FIGURA C.6.**




# Diagrama de componentes

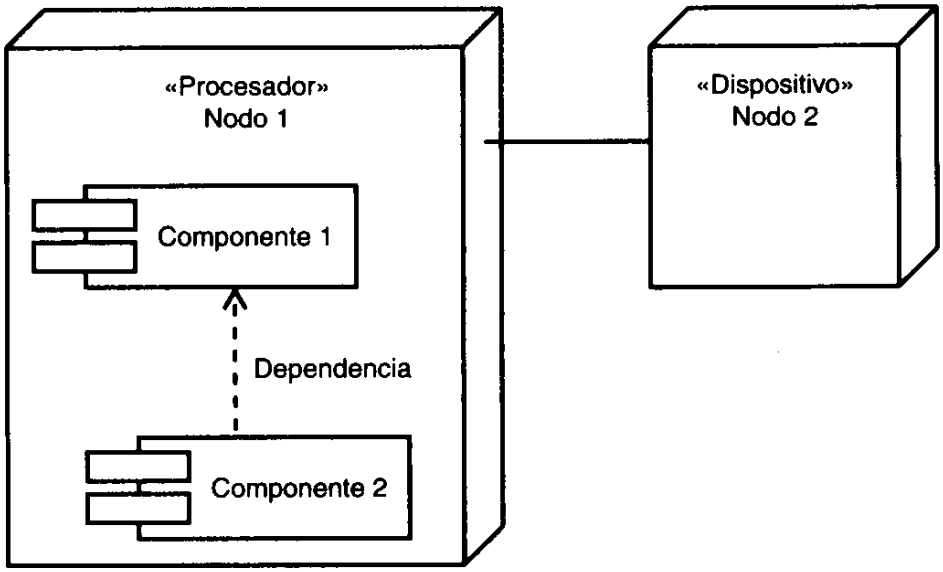
**FIGURA C.7**




www.elsolucionario.net

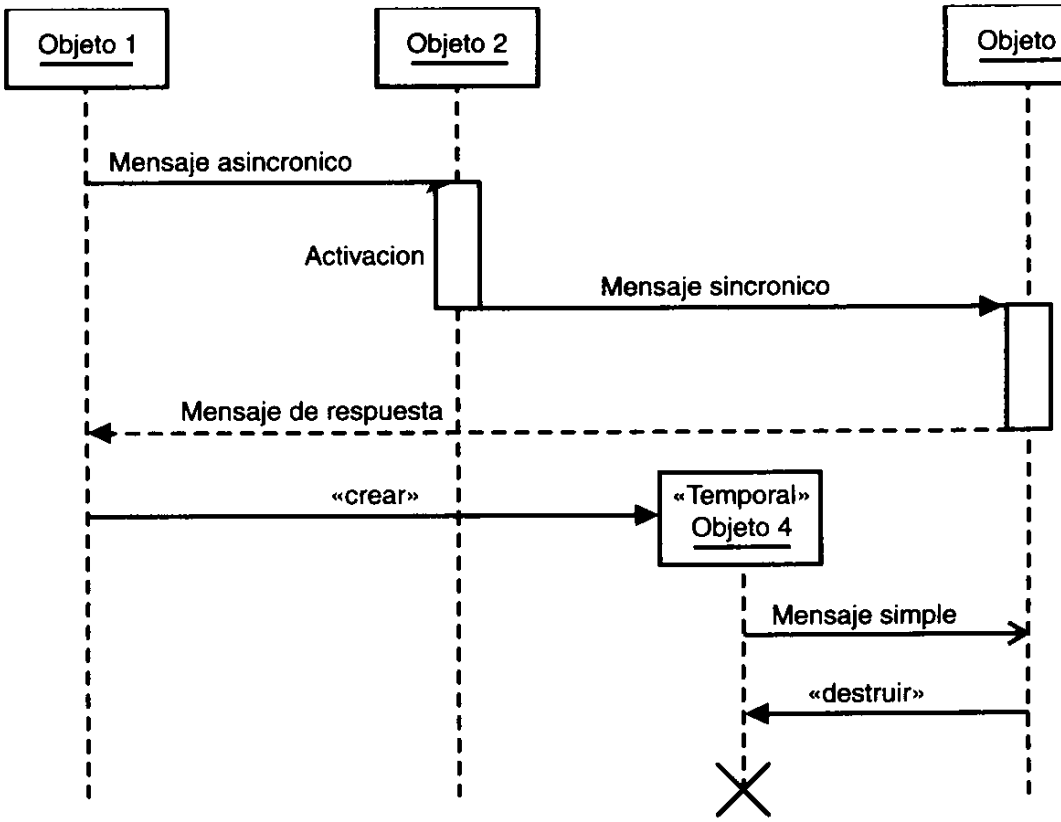
# Diagrama de distribución

**FIGURA C.8**




# Diagrama de secuencias

**FIGURA C.9**

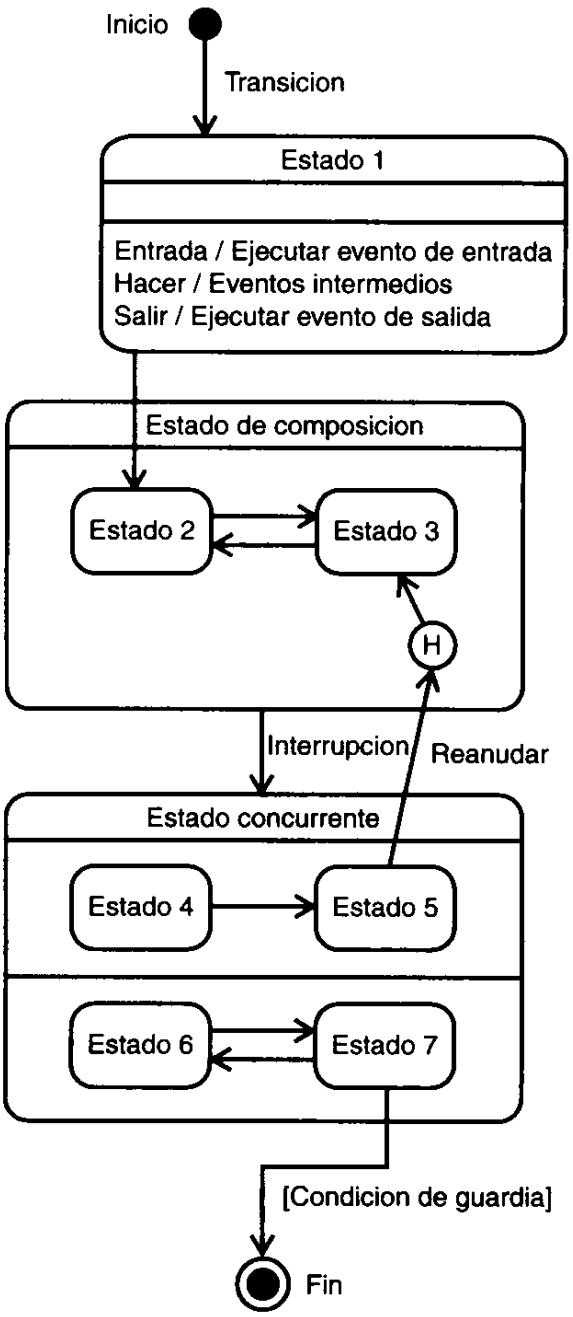



www.elsolucionario.net

www.elsolucionario.net

# Diagrama de estados

**FIGURA C.10**

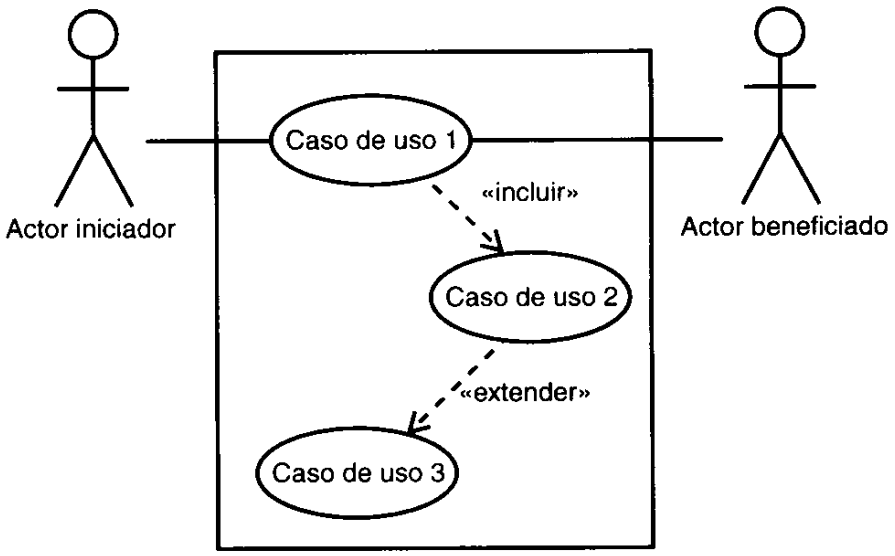



www.elsolucionario.net

www.elsolucionario.net

# Diagrama de casos de uso

**FIGURA C.11**




www.elsolucionario.net

www.elsolucionario.net