Referencia bibliográfica

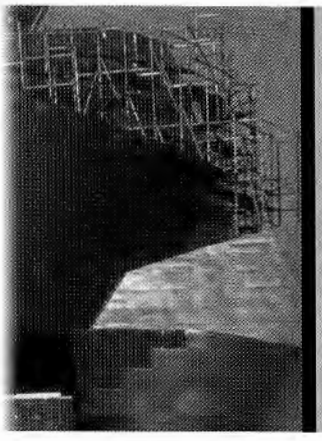
Sommerville, Ian. (2011). Modelado del sistema. En *Ingeniería de Software*. (Pp. 119-141). México: Pearson Educación.

SOMMERVILLE

INGENIERÍA DE SOFTWARE

9

PEARSON



INGENIERÍA DE SOFTWARE

Novena edición

Ian Sommerville

Traducción:
Víctor Campos Olguín
Traductor especialista en Sistemas Computacionales

Revisión técnica:
Sergio Fuenlabrada Velázquez
Edna Martha Miranda Chávez
Miguel Ángel Torres Durán
Mario Alberto Sesma Martínez
Mario Oviedo Galdeano
José Luis López Goytia
Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas-Instituto Politécnico Nacional, México

Darío Guillermo Cardacci
Universidad Abierta Interamericana, Buenos Aires, Argentina

Marcelo Martín Marciszack
Universidad Tecnológica Nacional, Córdoba, Argentina

Addison-Wesley

México • Argentina • Brasil • Colombia • Costa Rica • Chile • Ecuador
España • Guatemala • Panamá • Perú • Puerto Rico • Uruguay • Venezuela

Datos de catalogación bibliográfica

Sommerville, Ian

Ingeniería de Software

PEARSON EDUCACIÓN, México, 2011

ISBN: 978-607-32-0603-7

Área: Computación

Formato: 18.5 × 23.5 cm Páginas: 792

Edición en español

Editor: Luis M. Cruz Castillo
e-mail: luis.cruz@pearson.com
Editor de desarrollo: Felipe Hernández Carrasco
Supervisor de producción: Juan José García Guzmán

NOVENA EDICIÓN, 2011

ISBN VERSIÓN IMPRESA: 978-607-32-0603-7
ISBN VERSIÓN E-BOOK: 978-607-32-0604-4
ISBN E-CHAPTER: 978-607-32-0605-1

PRIMERA IMPRESIÓN

Impreso en México. *Printed in Mexico.*

1 2 3 4 5 6 7 8 9 0 - 14 13 12 11

Addison Wesley
es una marca de

PEARSON

Contenido breve

El modelado de sistemas es el proceso para desarrollar modelos abstractos de un sistema, donde cada modelo presenta una visión o perspectiva diferente de dicho sistema. En general, el modelado de sistemas se ha convertido en un medio para representar el sistema usando algún tipo de notación gráfica, que ahora casi siempre se basa en notaciones en el Lenguaje de Modelado Unificado (UML). Sin embargo, también es posible desarrollar modelos formales (matemáticos) de un sistema, generalmente como una especificación detallada del sistema. En este capítulo se estudia el modelado gráfico utilizando el UML, y en el capítulo 12, el modelado formal.

Los modelos se usan durante el proceso de ingeniería de requerimientos para ayudar a derivar los requerimientos de un sistema, durante el proceso de diseño para describir el sistema a los ingenieros que implementan el sistema, y después de la implementación para documentar la estructura y la operación del sistema. Es posible desarrollar modelos tanto del sistema existente como del sistema a diseñar:

1. Los modelos del sistema existente se usan durante la ingeniería de requerimientos. Ayudan a aclarar lo que hace el sistema existente y pueden utilizarse como base para discutir sus fortalezas y debilidades. Posteriormente, conducen a los requerimientos para el nuevo sistema.
2. Los modelos del sistema nuevo se emplean durante la ingeniería de requerimientos para ayudar a explicar los requerimientos propuestos a otros participantes del sistema. Los ingenieros usan tales modelos para discutir las propuestas de diseño y documentar el sistema para la implementación. En un proceso de ingeniería dirigido por modelo, es posible generar una implementación de sistema completa o parcial a partir del modelo del sistema.

El aspecto más importante de un modelo del sistema es que deja fuera los detalles. Un modelo es una abstracción del sistema a estudiar, y no una representación alternativa de dicho sistema. De manera ideal, una representación de un sistema debe mantener toda la información sobre la entidad a representar. Una abstracción simplifica y recoge deliberadamente las características más destacadas. Por ejemplo, en el muy improbable caso de que este libro se entregue por capítulos en un periódico, la presentación sería una abstracción de los puntos clave del libro. Si se tradujera del inglés al italiano, sería una representación alternativa. La intención del traductor sería mantener toda la información como se presenta en inglés.

Desde diferentes perspectivas, usted puede desarrollar diferentes modelos para representar el sistema. Por ejemplo:

1. Una perspectiva externa, donde se modelen el contexto o entorno del sistema.
2. Una perspectiva de interacción, donde se modele la interacción entre un sistema y su entorno, o entre los componentes de un sistema.
3. Una perspectiva estructural, donde se modelen la organización de un sistema o la estructura de datos que procese el sistema.
4. Una perspectiva de comportamiento, donde se modele el comportamiento dinámico del sistema y cómo responde ante ciertos eventos.

Estas perspectivas tienen mucho en común con la visión 4 + 1 de arquitectura del sistema de Kruchten (Kruchten, 1995), la cual sugiere que la arquitectura y la organización de un sistema deben documentarse desde diferentes perspectivas. En el capítulo 6 se estudia este enfoque 4 + 1.

En este capítulo se usan diagramas definidos en UML (Booch *et al.*, 2005; Rumbaugh *et al.*, 2004), que se han convertido en un lenguaje de modelado estándar para modelado orientado a objetos. El UML tiene numerosos tipos de diagramas y, por lo tanto, soporta la creación de muchos diferentes tipos de modelo de sistema. Sin embargo, un estudio en 2007 (Erickson y Siau, 2007) mostró que la mayoría de los usuarios del UML consideraban que cinco tipos de diagrama podrían representar lo esencial de un sistema.

1. Diagramas de actividad, que muestran las actividades incluidas en un proceso o en el procesamiento de datos.
2. Diagramas de caso de uso, que exponen las interacciones entre un sistema y su entorno.
3. Diagramas de secuencias, que muestran las interacciones entre los actores y el sistema, y entre los componentes del sistema.
4. Diagramas de clase, que revelan las clases de objeto en el sistema y las asociaciones entre estas clases.
5. Diagramas de estado, que explican cómo reacciona el sistema frente a eventos internos y externos.

Como aquí no hay espacio para discutir todos los tipos de diagramas UML, el enfoque se centrará en cómo estos cinco tipos clave de diagramas se usan en el modelado del sistema.

Cuando desarrolle modelos de sistema, sea flexible en la forma en que use la notación gráfica. No siempre necesitará apegarse rigurosamente a los detalles de una notación. El detalle y el rigor de un modelo dependen de cómo lo use. Hay tres formas en que los modelos gráficos se emplean con frecuencia:

1. Como medio para facilitar la discusión sobre un sistema existente o propuesto.
2. Como una forma de documentar un sistema existente.
3. Como una descripción detallada del sistema que sirve para generar una implementación de sistema.

En el primer caso, el propósito del modelo es estimular la discusión entre los ingenieros de software que intervienen en el desarrollo del sistema. Los modelos pueden ser incompletos (siempre que cubran los puntos clave de la discusión) y utilizar de manera informal la notación de modelado. Así es como se utilizan en general los modelos en el llamado "modelado ágil" (Ambler y Jeffries, 2002). Cuando los modelos se usan como documentación, no tienen que estar completos, pues quizás usted sólo desee desarrollar modelos para algunas partes de un sistema. Sin embargo, estos modelos deben ser correctos: tienen que usar adecuadamente la notación y ser una descripción precisa del sistema.

El Lenguaje de Modelado Unificado

El Lenguaje de Modelado Unificado es un conjunto compuesto por 13 diferentes tipos de diagrama para modelar sistemas de software. Surgió del trabajo en la década de 1990 sobre el modelado orientado a objetos, cuando anotaciones similares, orientadas a objetos, se integraron para crear el UML. Una amplia revisión (UML 2) se finalizó en 2004. El UML es aceptado universalmente como el enfoque estándar al desarrollo de modelos de sistemas de software. Se han propuesto variantes más generales para el modelado de sistemas.

http://www.SoftwareEngineering-9.com/Web/UML/

En el tercer caso, en que los modelos se usan como parte de un proceso de desarrollo basado en modelo, los modelos de sistema deben ser completos y correctos. La razón para esto es que se usan como base para generar el código fuente del sistema. Por lo tanto, debe ser muy cuidadoso de no confundir símbolos equivalentes, como las flechas de palo y las de bloque, que tienen significados diferentes.

Modelos de contexto

En una primera etapa en la especificación de un sistema, debe decidir sobre las fronteras del sistema. Esto implica trabajar con los participantes del sistema para determinar cuál funcionalidad se incluirá en el sistema y cuál la ofrece el entorno del sistema. Tal vez decida que el apoyo automatizado para algunos procesos empresariales deba implementarse, pero otros deben ser procesos manuales o soportados por sistemas diferentes. Debe buscar posibles traslapes en la funcionalidad con los sistemas existentes y determinar dónde tiene que implementarse nueva funcionalidad. Estas decisiones deben hacerse oportunamente durante el proceso, para limitar los costos del sistema, así como el tiempo necesario para comprender los requerimientos y el diseño del sistema.

En algunos casos, la frontera entre un sistema y su entorno es relativamente clara. Por ejemplo, donde un sistema automático sustituye un sistema manual o computarizado, el entorno del nuevo sistema, por lo general, es el mismo que el entorno del sistema existente. En otros casos, existe más flexibilidad y usted es quien decide qué constituye la frontera entre el sistema y su entorno, durante el proceso de ingeniería de requerimientos.

Por ejemplo, imagine que desarrolla la especificación para el sistema de información de pacientes para atención a la salud mental. Este sistema intenta manejar la información sobre los pacientes que asisten a clínicas de salud mental y los tratamientos que les prescriben. Al desarrollar la especificación para este sistema, debe decidir si el sistema tiene que enfocarse exclusivamente en reunir información de las consultas (junto con otros sistemas para recopilar información personal acerca de los pacientes), o si también es necesario que recopile datos personales acerca del paciente. La ventaja de apoyarse en otros sistemas para la información del paciente es que evita duplicar datos. Sin embargo, la principal desventaja es que usar otros sistemas haría más lento el acceso a la información. Si estos sistemas no están disponibles, entonces no pueden usarse en MHC-PMS.

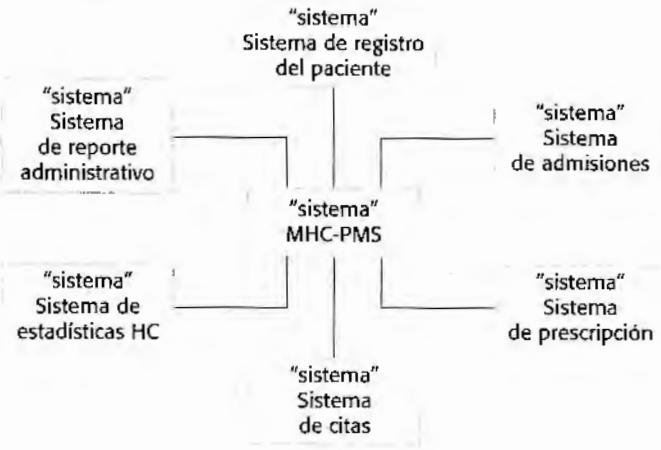


Figura 5.1 El contexto del MHC-PMS

La definición de frontera de un sistema no es un juicio libre de valor. Las preocupaciones sociales y organizacionales pueden significar que la posición de la frontera de un sistema se determine considerando factores no técnicos. Por ejemplo, una frontera de sistema puede colocarse deliberadamente, de modo que todo el proceso de análisis se realice en un sitio; puede elegirse de forma que sea innecesario consultar a un administrador particularmente difícil; puede situarse de manera que el costo del sistema aumente y la división de desarrollo del sistema deba, por lo tanto, expandirse al diseño y la implementación del sistema.

Una vez tomadas algunas decisiones sobre las fronteras del sistema, parte de la actividad de análisis es la definición de dicho contexto y las dependencias que un sistema tiene con su entorno. Normalmente, producir un modelo arquitectónico simple es el primer paso en esta actividad.

La figura 5.1 es un modelo de contexto simple que muestra el sistema de información del paciente y otros sistemas en su entorno. A partir de la figura 5.1, se observa que el MHC-PMS está conectado con un sistema de citas y un sistema más general de registro de pacientes, con el cual comparte datos. El sistema también está conectado a sistemas para manejo de reportes y asignación de camas de hospital, y un sistema de estadísticas que recopila información para la investigación. Finalmente, utiliza un sistema de prescripción que elabora recetas para la medicación de los pacientes.

Los modelos de contexto, por lo general, muestran que el entorno incluye varios sistemas automatizados. Sin embargo, no presentan los tipos de relaciones entre los sistemas en el entorno y el sistema que se especifica. Los sistemas externos generan datos para el sistema o consumen datos del sistema. Pueden compartir datos con el sistema, conectarse directamente, a través de una red, o no conectarse en absoluto. Pueden estar físicamente juntos o ubicados en edificios separados. Todas estas relaciones llegan a afectar los requerimientos y el diseño del sistema a definir, por lo que deben tomarse en cuenta.

Por consiguiente, los modelos de contexto simples se usan junto con otros modelos, como los modelos de proceso empresarial. Éstos describen procesos humanos y automatizados que se usan en sistemas particulares de software.

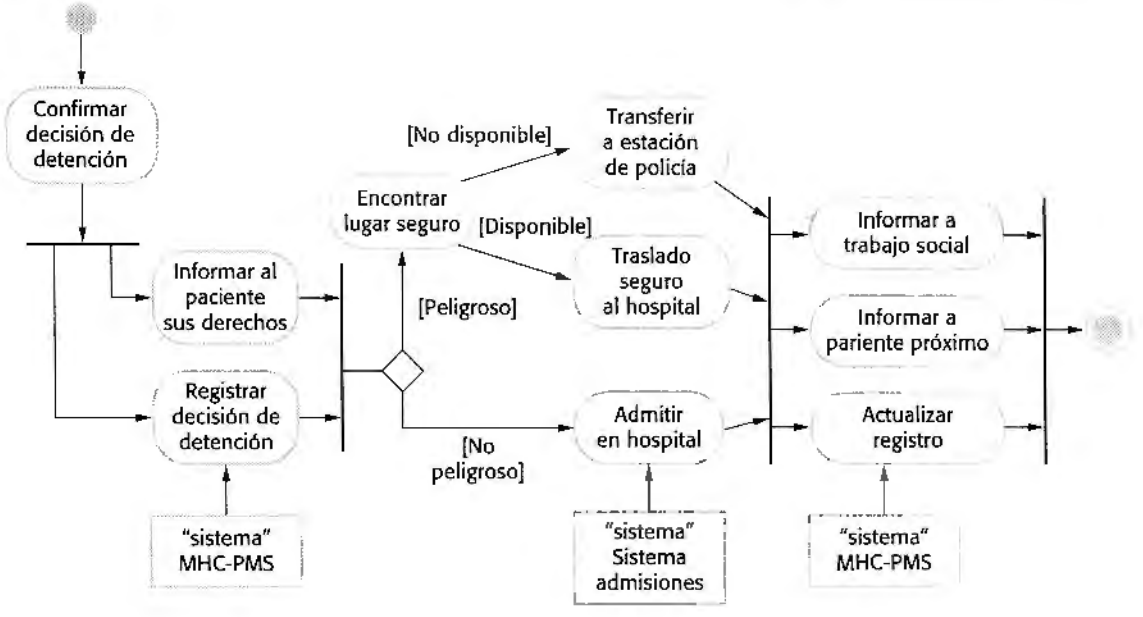


Figura 5.2 Modelo del proceso de detención involuntaria

La figura 5.2 es un modelo de un importante proceso de sistema que muestra los procesos en que se utiliza el MHC-PMS. En ocasiones. los pacientes que snfren de problemas de salud mental son un riesgo para otros o para sí mismos. Por ello, es posible qne en un hospital deban mantenerse contra su voluntad para que se les suministre el tratamiento. Tal detención está sujeta a estrictas protecciones legales, por ejemplo, la decisión de detener a un paciente tiene qne revisarse con regularidad, para que no se detenga a la persona indefinidamente sin una buena razón. Una de las funciones del MHC-PMS es garantizar que se implementen dichas protecciones.

La figura 5.2 es un diagrama de actividad UML. Los diagramas de actividad intentan mostrar las actividades que inclnyen un proceso de sistema, así como el flujo de control de una actividad a otra. El inicio de un proceso se indica con un círculo lleno; el fin, mediante un círculo lleno dentro de otro círculo. Los rectángulos con esqninas redondeadas representan actividades, esto es, los subprocesos específicos qne hay que realizar. Puede incluir objetos en los gráficos de actividad. En la fignra 5.2 se muestran los sistemas que sirven para apoyar diferentes procesos. Se indicó que éstos son sistemas separados al usar la característica de estereotipo UML.

En un diagrama de actividad UML, las flechas representan el flujo de trabajo de una actividad a otra. Una barra sólida se emplea para indicar coordinación de actividades. Cuando el flujo de más de una actividad se dirige a una barra sólida, entonces todas esas actividades deben completarse antes del posible avance. Cuando el flujo de una barra sólida conduzca a algunas actividades, éstas pueden ejecutarse en forma paralela. Por consiguiente, en la figura 5.2, las actividades para informar a trabajo social y al familiar cercano del paciente, así como para actualizar el registro de detención, pueden ser concurrentes.

Las flechas pueden anotarse con guardas que indiquen la condición al tomar dicho flujo. En la figura 5.2 se observan guardas que muestran los flujos para pacientes que son

un riesgo para la sociedad y quienes no lo son. Los pacientes peligrosos deben mantenerse en una instalación segura. No obstante, los pacientes suicidas que, por lo tanto, representan un riesgo para sí mismos, se detendrían en un pabellón hospitalario adecuado.

5.2 Modelos de interacción

Todos los sistemas incluyen interacciones de algún tipo. Éstas son interacciones del usuario, que implican entradas y salidas del usuario; interacciones entre el sistema a desarrollar y otros sistemas; o interacciones entre los componentes del sistema. El modelado de interacción del usuario es importante, pues ayuda a identificar los requerimientos del usnario. El modelado de la interacción sistema a sistema destaca los problemas de comunicación que se lleguen a presentar. El modelado de interacción de componentes ayuda a entender si es probable que una estructura de un sistema propuesto obtenga el rendimiento y la confiabilidad requeridos por el sistema.

En esta sección se cubren dos enfoques relacionados con el modelado de interacción:

1. Modelado de caso de uso, que se utiliza principalmente para modelar interacciones entre un sistema y actores externos (usuarios u otros sistemas).
2. Diagramas de secuencia, que se emplean para modelar interacciones entre componentes del sistema, aunque también pueden incluirse agentes externos.

Los modelos de caso de uso y los diagramas de secuencia presentan la interacción a diferentes niveles de detalle y, por lo tanto, es posible utilizarlos juntos. Los detalles de las interacciones que hay en un caso de uso de alto nivel se documentan en un diagrama de secuencia. El UML también incluye diagramas de comunicación usados para modelar interacciones. Aquí no se analiza esto, ya que se trata de representaciones alternativas de gráficos de secuencia. De hecho, algunas herramientas pueden generar un diagrama de comunicación a partir de nn diagrama de secuencia.

5.2.1 Modelado de casos de uso

El modelado de casos de uso fue desarrollado originalmente por Jacobson y sus colaboradores (1993) en la década de 1990, y se incorporó en el primer lanzamiento del UML (Rumbaugh *et al.*, 1999). Como se estudió en el capítulo 4, el modelado de casos de uso se utiliza ampliamente para apoyar la adquisición de requerimientos. Un caso de uso puede tomarse como un simple escenario que describa lo que espera el usuario de un sistema.

Cada caso de uso representa una tarea discreta qne implica interacción externa con un sistema. En su forma más simple, un caso de uso se muestra como una elipse, con los actores que intervienen en el caso de uso representados como figuras humanas. La figura 5.3 presenta nn caso de uso del MHC-PMS que implica la tarea de subir datos desde el MHC-PMS hasta un sistema más general de registro de pacientes. Este sistema más general mantiene un resumen de datos sobre el paciente, en vez de los datos sobre cada consulta, que se registran en el MHC-PMS.

Figura 5.3 Caso de uso de transferencia de datos

Observe que en este caso de uso hay dos actores: el operador qne transfiere los datos y el sistema de registro de pacientes. La notación con fignra humana se desarrolló originalmente para cubrir la interacción entre individuos, pero también se usa ahora para representar otros sistemas externos y el hardware. De manera formal, los diagramas de caso de uso deben emplear líneas sin flechas; las flechas en el UML indican la dirección del flnjo de mensajes. Evidentemente, en un caso de uso los mensajes pasan en ambas direcciones. Sin embargo, las flechas en la figura 5.3 se usan de manera informal para indicar qne la recepcionista médica inicia la transacción y los datos se transfieren al sistema de registro de pacientes.

Los diagramas de caso de uso brindan un panorama bastante sencillo de una interacción, de modo que usted tieue que ofrecer más detalle para entender lo que está implicado. Este detalle puede ser una simple descripciou textual, o una descripción estructurada en una tabla o un diagrama de secuencia, como se discute a continuación. Es posible elegir el formato más adecuado, dependiendo del caso de uso y del nivel de detalle que usted considere se requiera en el modelo. Para el autor, el formato más útil es un formato tabular estándar. La figura 5.4 ilustra una descripción tabular del caso de uso "transferencia de datos".

Como vimos en el capítulo 4, los diagramas de caso de uso compuestos indican un número de casos de uso diferentes. En ocasiones, se incluyen todas las interacciones posibles con nn sistema en un solo diagrama de caso de uso compuesto. Sin embargo, esto qnizá sea imposible debido a la cantidad de casos de uso. En tales situaciones, pnede desarrollar varios diagramas, cada uno de los cuales exponga casos de nso relacionados. Por ejemplo, la figura 5.5 presenta todos los casos de nso en el MHC-PMS, en los cnales interviene el actor "recepcionista médico".

Figura 5.4 Descripción tabular del caso de uso "transferencia de datos"

MHC-PMS: Transferencia de datos	
Actores	Recepcionista médico, sistema de registros de paciente (PRS).
Descripción	Un recepcionista puede transferir datos del MHC-PMS a una base de datos general de registro de pacientes, mantenida por una autoridad sanitaria. La información transferida puede ser información personal actualizada (dirección, número telefónico, etc.) o un resumen del diagnóstico y tratamiento del paciente.
Datos	Información personal del paciente, resumen de tratamiento.
Estímulo	Comando de usuario emitido por recepcionista médico.
Respuesta	Confirmación de que el PRS se actualizó.
Comentarios	El recepcionista debe tener permisos de seguridad adecuados para acceder a la información del paciente y al PRS.

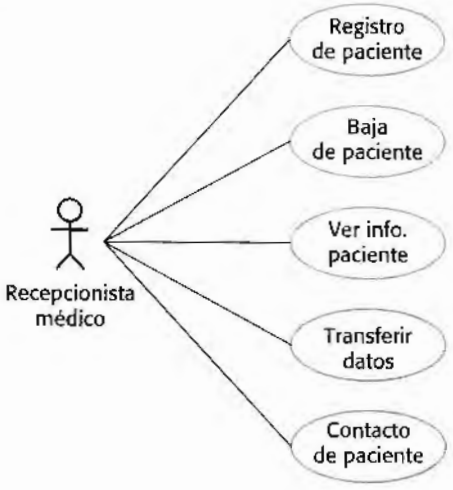


Figura 5.5 Casos de uso que involucran el papel "recepcionista médico"

5.2.2 Diagramas de secuencia

Los diagramas de secuencia en el UML se usan principalmente para modelar las interacciones entre los actores y los objetos en un sistema, así como las interacciones entre los objetos en sí. El UML tiene una amplia sintaxis para diagramas de secuencia, lo cual permite muchos tipos diferentes de interacción a modelar. Como aqní no hay espacio para cubrir todas las posibilidades, sólo nos enfocaremos en lo básico de este tipo de diagrama.

Como sugiere el nombre, un diagrama de secuencia muestra la sucesión de interacciones qne ocurre dnrante un caso de uso particular o una instancia de caso de uso. La figura 5.6 es un ejemplo de un diagrama de secuencia que ilustra los fundamentos de la notación. Estos modelos de diagrama incluyen las interacciones en el caso de uso "ver información de paciente", donde un recepcionista médico puede conocer la información de algún paciente.

Los objetos y actores que intervienen se mencionan a lo largo de la parte superior del diagrama, con una línea punteada que se dibuja verticalmente a partir de éstos. Las interacciones entre los objetos se indican con flechas dirigidas. El rectángulo sobre las líneas punteadas indica la línea de vida del objeto tratado (es decir, el tiempo que la instancia del objeto está involucrada en la computación). La secuencia de interacciones se lee de arriba abajo. Las anotaciones sobre las flechas señalan las llamadas a los objetos, sus parámetros y los valores que regresan. En este ejemplo, también se muestra la notación empleada para exponer alternativas. Un recuadro marcado con "alt" se usa con las condiciones indicadas entre corchetes.

La figura 5.6 se lee del signiente modo:

1. El recepcionista médico activa el método ViewInfo (ver información) en una instancia P de la clase de objeto PatientInfo, y suministra el identificador del paciente, PID. P es un objeto de interfaz de usuario, que se despliega como un formato que muestra la información del paciente.

2. La instancia P llama a la base de datos para regresar la información requerida, y suministra el identificador del recepcionista para permitir la verificación de seguridad (en esta etapa no se preocupe de dónde proviene este UID).

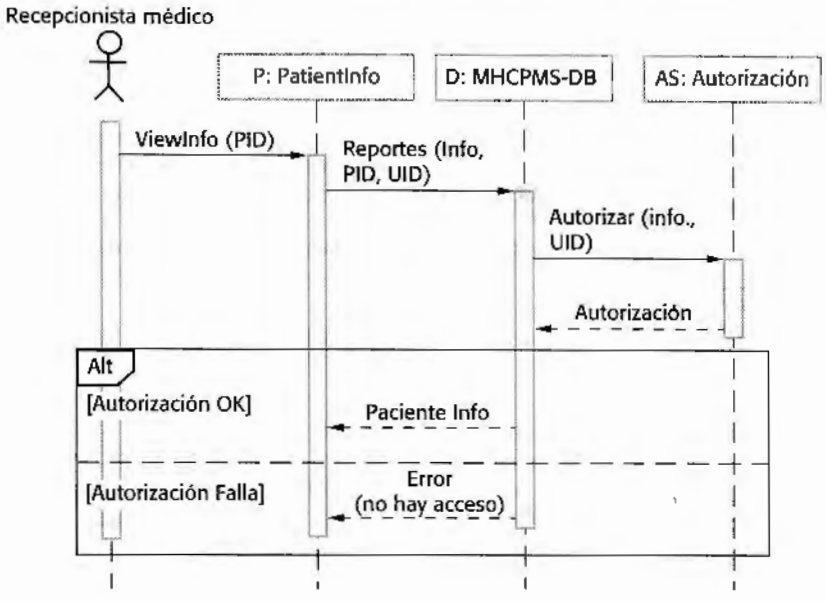


Figura 5.6 Diagrama de secuencia para "ver información del paciente"

3. La base de datos comprueba, mediante un sistema de autorización, que el usuario esté autorizado para tal acción.
4. Si está autorizado, se regresa la información del paciente y se llena un formato en la pantalla del usuario. Si la autorización falla, entonces se regresa un mensaje de error.

La figura 5.7 es un segundo ejemplo de nn diagrama de secuencia del mismo sistema que ilustra dos características adicionales. Se trata de la comunicación directa entre los actores en el sistema y la creación de objetos como parte de una secuencia de operaciones. En este ejemplo, un objeto del tipo Summary (resumen) se crea para contener los datos del resumen que deben subirse al PRS (*patient record system*, es decir, el sistema de registro de paciente). Este diagrama se lee de la signiente manera:

1. El recepcionista inicia sesión (log) en el PRS.
2. Hay dos opciones disponibles. Las opciones permiten la transferencia directa de información actualizada del paciente al PRS, y la transferencia de datos del resumen de salud del MHC-PMS al PRS.
3. En cada caso, se verifican los permisos del recepcionista usando el sistema de autorización.
4. La información personal se transfiere directamente del objeto de interfaz del usuario al PRS. De manera alternativa, es posible crear un registro del resumen de la base de datos y, luego, transferir dicho registro.
5. Al completar la transferencia, el PRS emite un mensaje de estatus y el usuario termina la sesión (log off).

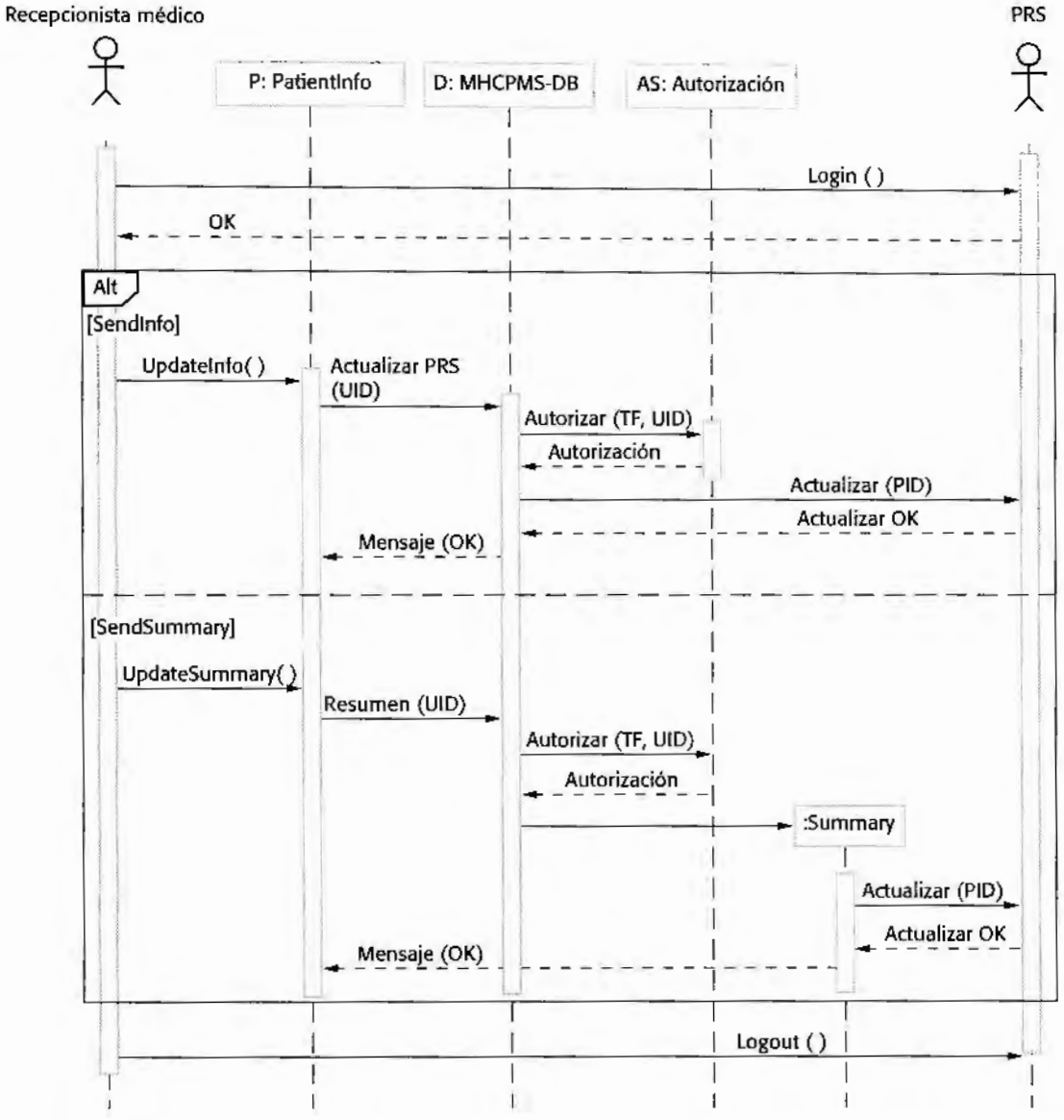


Figura 5.7 Diagrama de secuencia para transferir datos

A menos que use diagramas de secuencia para generación de código o documentación detallada, en dichos diagramas no tiene que incluir todas las interacciones. Si desarrolla modelos iniciales de sistema en el proceso de desarrollo para apoyar la ingeniería de requerimientos y el diseño de alto nivel, habrá muchas interacciones que dependan de decisiones de implementación. Por ejemplo, en la figura 5.7, la decisión sobre cómo conseguir el identificador del usuario para comprobar la autorización podría demorarse. En una implementación, esto implicaría la interacción con un objeto User (usuario), pero esto no es importante en esta etapa y, por lo tanto, no necesita incluirse en el diagrama de secuencia.

Análisis de requerimientos orientado a objetos

En el análisis de requerimientos orientado a objetos, se modelan entidades del mundo real usando clases de objetos. Usted puede crear diferentes tipos de modelos de objetos, que muestren cómo se relacionan mutuamente las clases de objetos, cómo se agregan objetos para formar otros objetos, cómo interactúan los objetos entre sí, etcétera. Cada uno de éstos presenta información única acerca del sistema que se especifica.

http://www.SoftwareEngineering-9.com/Web/OORA/

5.3 Modelos estructurales

Los modelos estructurales de software muestran la organización de un sistema, en términos de los componentes que constituyen dicho sistema y sus relaciones. Los modelos estructurales son modelos estáticos, que muestran la estructura del diseño del sistema, o modelos dinámicos, que revelan la organización del sistema cuando se ejecuta. No son lo mismo: la organización dinámica de un sistema como un conjunto de hilos en interacción tiende a ser mny diferente de un modelo estático de componentes del sistema.

Los modelos estructurales de un sistema se crean cuando se discute y diseña la arquitectura del sistema. El diseño arquitectónico es un tema particularmente importante en la ingeniería de software, y los diagramas UML de componente, de paquete y de implementación se utilizan cuando se presentan modelos arquitectónicos. En los capítulos 6, 18 y 19 se cubren diferentes aspectos de la arquitectura de software y del tuodelado arquitectónico. Esta sección se enfoca en el uso de diagratuas de clase para modelar la estructura estática de las clases de objetos, en un sistema de software.

5.3.1 Diagramas de clase

Los diagramas de clase pueden usarse cuando se desarrolla un modelo de sistema orientado a objetos para mostrar las clases en un sistema y las asociaciones entre dichas clases. De manera holgada, una clase de objeto se considera como una definición general de un tipo de objeto del sistema. Una asociación es un vínculo entre clases, que indica que hay una relación entre dicha clases. En consecuencia, cada clase puede tener algún conocimiento de esta clase asociada.

Cuando se desarrollan modelos durante las primeras etapas del proceso de ingeniería de software, los objetos representan algo en el mundo real, como un paciente, una receta, un médico, etcétera. Conforme se desarrolla una implementación, por lo general se necesitan definir los objetos de implementación adicionales que se usan para dar la funcionalidad requerida del sistema. Aqní, el enfoque está sobre el modelado de objetos del mundo real, como parte de los requerimientos o los primeros procesos de diseño del software.

Los diagramas de clase en el UML pueden expresarse con diferentes niveles de detalle. Cuando se desarrolla un modelo, la primera etapa con frecnencia implica buscar en el mundo, identificar los objetos esenciales y representarlos como clases. La forma más sencilla de hacer esto es escribir el nombre de la clase en un recuadro. También puede anotar la existencia de una asociación dibnjando simplemente nna línea entre las clases.

Figura 5.8 Clases y asociación UML

Por ejemplo, la figura 5.8 es un diagrama de clase simple que muestra dos clases: Patient (paciente) y Patient Record (registro del paciente), con una asociación entre ellos.

En la figura 5.8 se ilustra una característica más de los diagramas de clase: la habilidad para mostrar cuántos objetos intervienen en la asociación. En este ejemplo, cada extremo de la asociación se registra con un 1, lo cual significa que hay una relación 1:1 entre objetos de dichas clases. Esto es, cada paciente tiene exactamente un registro, y cada registro conserva información precisa del paciente. En los últimos ejemplos se observa que son posibles otras multiplicidades. Se define un número exacto de objetos que están implicados, o bien, con el uso de un asterisco (*), como se muestra en la figura 5.9, que hay un número indefinido de objetos en la asociación.

La figura 5.9 desarrolla este tipo de diagrama de clase para mostrar que los objetos de la clase "paciente" también intervienen en relaciones con varias otras clases. En este ejemplo, se observa que es posible nombrar las asociaciones para dar al lector un indicio del tipo de relación que existe. Asimismo, el UML permite especificar el papel de los objetos que participan en la asociación.

En este nivel de detalle, los diagramas de clase parecen modelos semánticos de datos. Los modelos semánticos de datos se usan en el diseño de bases de datos. Muestran las entidades de datos, sus atributos asociados y las relaciones entre dichas entidades. Este enfoque para modelar fue propuesto por primera vez por Chen (1976), a mediados de la década de 1970; desde entonces, se han desarrollado diversas variantes (Codd, 1979; Hammer y McLeod, 1981; Hull y King, 1987), todas con la misma forma básica.

El UML no incluye una notación específica para este modelado de bases de datos, ya que supone un proceso de desarrollo orientado a objetos, así como modelos de datos que usan objetos y sus relaciones. Sin embargo, es posible usar el UML para representar un modelo semántico de datos. En un modelo semántico de datos, piense en entidades como

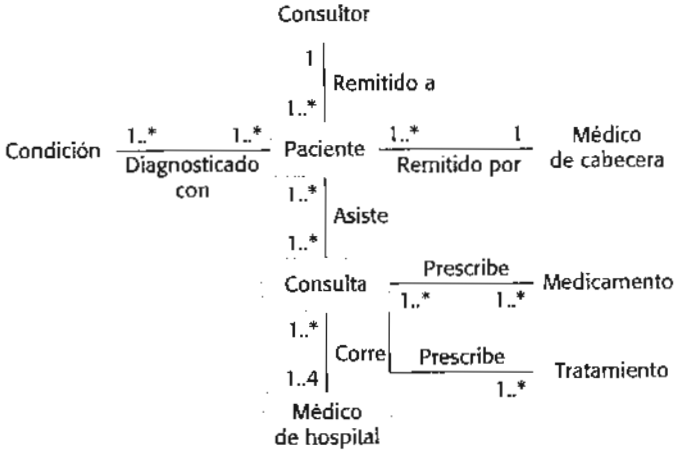


Figura 5.9 Clases y asociaciones en el MHC-PMS

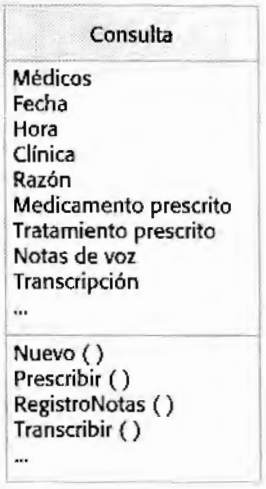


Figura 5.10 La clase de consulta

clases de objeto simplificadas (no tienen operaciones), atribntos como atributos de clase de objeto y relaciones como nombres de asociaciones entre clases de objeto.

Cuando se muestran las asociaciones entre clases, es conveniente representar dichas clases en la forma más sencilla posible. Para definirlas con más detalle, agregue información sobre sus atribntos (las características de un objeto) y operaciones (aquello que se puede solicitar de un objeto). Por ejemplo, un objeto Patient tendrá el atributo Address (dirección) y puede incluir una operación llamada ChangeAddress (cambiar dirección), que se llama cuando nn paciente manifiesta que se mudó de una dirección a otra. En el UML, los atributos y las operaciones se muestran al extender el rectángulo simple qne representa una clase. Esto se ilustra en la figura 5.10, donde:

1. El nombre de la clase de objeto está en la sección superior.
2. Los atributos de clase están en la sección media. Esto debe inclnir los nombres del atributo y, opcionalmente, sns tipos.
3. Las operaciones (llamadas métodos en Java y en otros lenguajes de programación OO) asociadas con la clase de objeto están en la sección inferior del rectángulo.

La fignra 5.10 expone posibles atributos y operaciones sobre la clase Consnlta (Consultation). En este ejemplo, se snpone que los médicos registran notas de voz qne se transcriben más tarde para registrar detalles de la consulta. Al prescribir fármacos, el médico debe usar el nuétodo Prescribir (Prescribe) para generar una receta electrónica.

5.3.2 Generalización

La generalización es una técnica cotidiana que se usa para gestionar la complejidad. En vez de aprender las características detalladas de cada entidad que se experimenta, dichas entidades se colocan en clases más generales (animales, automóviles, casas,

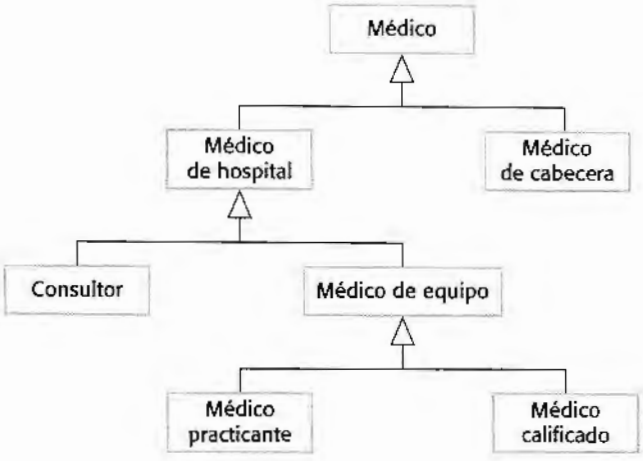


Figura 5.11 Jerarquía de generalización

etcétera) y se aprenden las características de dichas clases. Esto permite deducir que diferentes miembros de estas clases tienen algunas características comnes (por ejemplo, las ardillas y ratas son roedores). Es posible hacer enunciados generales que se apliquen a todos los miembros de la clase (por ejemplo, todos los roedores tienen dientes para roer).

En el modelado de sistemas, con frecnencia es útil examinar las clases en un sistema, con la finalidad de ver si hay ámbito para la generalización. Esto significa que la información común se mantendrá solamente en nn lugar. Ésta es una buena práctica de diseño, pnes significa que, si se proponen cambios, entonces no se tiene que buscar en todas las clases en el sistema, para observar si se ven afectadas por el cambio. En los lenguajes orientados a objetos, como Java, la generalización se implementa usando los mecanismos de herencia de clase construidos en el lenguaje.

El UML tiene un tipo específico de asociación para denotar la generalización, como se ilustra en la figura 5.11. La generalización se muestra como una flecha que apunta hacia la clase más general. Esto indica que los médicos de cabecera y los médicos de hospital pueden generalizarse como médicos, y que hay tres tipos de médicos de hospital: quienes se graduaron recientemente de la escuela de medicina y tienen que ser supervisados (médicos practicantes); quienes trabajan sin supervisión como parte de un equipo de consultores (médicos registrados); y los consultores, que son médicos experimentados con plenas responsabilidades en la toma de decisiones.

En una generalización, los atributos y las operaciones asociados con las clases de nivel superior también se asocian con las clases de nivel inferior. En esencia, las clases de nivel inferior son subclases que heredan los atributos y las operaciones de sus superclases. Entonces dichas clases de nivel inferior agregan atributos y operaciones más específicos. Por ejemplo, todos los médicos tienen un nombre y número telefónico; todos los médicos de hospital tienen un número de personal y un departamento, pero los médicos de cabecera no tienen tales atributos, pues trabajan de manera independiente. Sin embargo, sí tienen un nombre de consultorio y dirección. Esto se ilustra en la figura 5.12, que muestra parte de la jerarquía de generalización que se extendió con atributos de clase. Las operaciones asociadas con la clase "médico" buscan registrar y dar de baja al médico con el MHC-PMS.

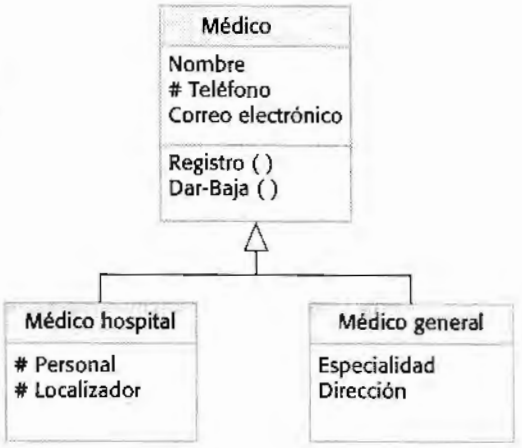


Figura 5.12 Jerarquía de generalización con detalles agregados

5.3.3 Agregación

Los objetos en el mundo real con frecnencia están compuestos por diferentes partes. Un paqnete de estudio para un curso, por ejemplo, estaría compuesto por libro, diapositivas de PowerPoint, exámenes y recomendaciones para lecturas posteriores. En ocasiones, en un modelo de sistema, nsted necesita ilustrar esto. El UML proporciona un tipo especial de asociación entre clases llamado agregación, que significa que un objeto (el todo) se compone de otros objetos (las partes). Para mostrarlo, se usa un trazo en forma de diamante, junto con la clase que representa el todo. Esto se ilustra en la figura 5.13, que indica que un registro de paciente es nna composición de Paciente (Patient) y un número indefinido de Consnlta (Consultations).

5.4 Modelos de comportamiento

Los modelos de comportamiento son modelos dinámicos del sistema conforme se ejecutan. En ellos se muestra lo que sucede o lo que se supone que pasa cuando un sistema responde ante un estímnlo de su entorno. Tales estímulos son de dos tipos:

1. *Datos* Algnnos datos que llegan se procesan por el sistema.
2. *Eventos* Algunos eventos activan el procesamiento del sistema. Los eventos pneden tener datos asociados, pero esto no es siempre el caso.

Figura 5.13 La asociación agregación

Diagramas de flujo de datos

Los diagramas de flujo de datos (DFD) son modelos de sistema que presentan una perspectiva funcional, donde cada transformación constituye una sola función o un solo proceso. Los DFD se usan para mostrar cómo fluyen los datos a través de una secuencia de pasos del procesamiento. Por ejemplo, un paso del procesamiento sería el filtrado de registros duplicados en una base de datos de clientes. Los datos se transforman en cada paso antes de moverse hacia la siguiente etapa. Dichos pasos o transformaciones del procesamiento representan procesos o funciones de software, en los cuales los diagramas de flujo de datos se usan para documentar un diseño de software.

http://www.SoftwareEngineering-9.com/Web/DFDs

Muchos sistemas empresariales son sistemas de procesamiento de datos que se activan principalmente por datos. Son controlados por la entrada de datos al sistema con relativamente poco procesamiento externo de eventos. Su procesamiento incluye una secuencia de acciones sobre dichos datos y la generación de una salida. Por ejemplo, un sistema de facturación telefónica aceptará información de las llamadas hechas por un cliente, calculará los costos de dichas llamadas y generará una factura para enviarla a dicho cliente. En contraste, los sistemas de tiempo real muchas veces están dirigidos por un evento con procesamiento de datos mínimo. Por ejemplo, un sistema de conmutación telefónico terrestre responde a eventos como "receptor ocupado" al generar un tono de dial, o al presionar las teclas de un teléfono para la captura del número telefónico, etcétera.

5.4.1 Modelado dirigido por datos

Los modelos dirigidos por datos muestran la secuencia de acciones involucradas en el procesamiento de datos de entrada, así como la generación de una salida asociada. Son particularmente útiles durante el análisis de requerimientos, pues sirven para mostrar procesamiento "extremo a extremo" en un sistema. Esto es, exhiben toda la secuencia de acciones que ocurren desde una entrada a procesar hasta la salida correspondiente, que es la respuesta del sistema.

Los modelos dirigidos por datos están entre los primeros modelos gráficos de software. En la década de 1970, los métodos estructurados como el análisis estructurado de DeMarco (DeMarco, 1978) introdujeron los diagramas de flujo de datos (DFD), como una forma de ilustrar los pasos del procesamiento en un sistema. Los modelos de flujo de datos son útiles porque el hecho de rastrear y documentar cómo los datos asociados con un proceso particular se mueven a través del sistema ayuda a los analistas y diseñadores a entender lo que sucede. Los diagramas de flujo de datos son simples e intuitivos y, por lo general, es posible explicarlos a los usuarios potenciales del sistema, quienes después pueden participar en la validación del modelo.

El UML no soporta diagramas de flujo de datos, puesto que originalmente se propusieron y usaron para modelar el procesamiento de datos. La razón para esto es que los DFD se enfocan en funciones del sistema y no reconocen objetos del sistema. Sin embargo, como los sistemas dirigidos por datos son tan comunes en los negocios, UML 2.0 introdujo diagramas de actividad, que son similares a los diagramas de flujo de datos. Por ejemplo, la figura 5.14 indica la cadena de procesamiento involucrada en el software de la bomba de insulina. En este diagrama, se observan los pasos de procesamiento (representados como actividades) y los datos que fluyen entre dichos pasos (representados como objetos).

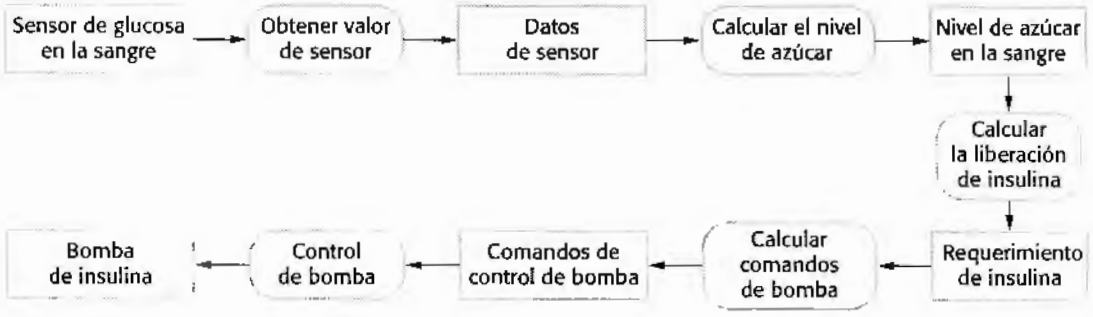


Figura 5.14 Modelo de actividad de la operación de la bomba de insulina

Una forma alternativa de mostrar la secuencia de procesamiento en un sistema es usar diagramas de secuencia UML. Ya se vio cómo se utilizan para modelar interacción pero, si los utiliza para que dichos mensajes sólo se envíen de izquierda a derecha, luego muestran el procesamiento secuencial de datos en el sistema. La figura 5.15 ilustra esto, con un modelo de secuencia del procesamiento de un pedido y envío a un proveedor. Los modelos de secuencia destacan los objetos en un sistema, mientras que los diagramas de flujo de datos resaltan las funciones. El diagrama de flujo de datos equivalente para la orden de procesamiento se incluye en las páginas Web del libro.

5.4.2 Modelado dirigido por un evento

El modelado dirigido por un evento muestra cómo responde un sistema a eventos externos e internos. Se basa en la suposición de que un sistema tiene un número finito de estados y que los eveutos (estímulos) pueden causar una transición de un estado a otro. Por ejemplo, un sistema que controla una válvula puede moverse de un estado de "válvula abierta" a un estado de "válvula cerrada", cuando recibe un comando operador (el estímulo). Esta visión de un sistema es adecuado particularmente para sistema en tiempo real. El modelado basado en eventos se introdujo en los métodos de diseño en tiempo real, como los propuestos por Ward y Mellor (1985) y Harel (1987, 1988).

El UML soporta modelado basado en eventos usando diagramas de estado, que se fundamentaron en gráficos de estado (Harel, 1987, 1988). Los diagramas de estado muestran estados y eventos del sistema que causan transiciones de un estado a otro. No exponen el

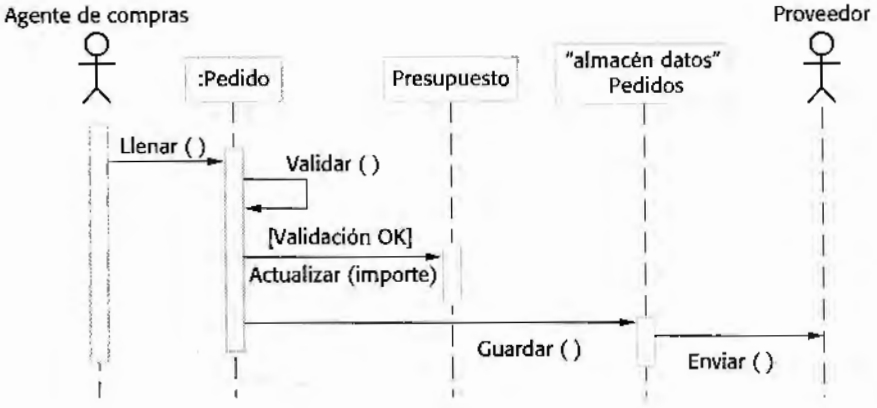


Figura 5.15 Orden de procesamiento

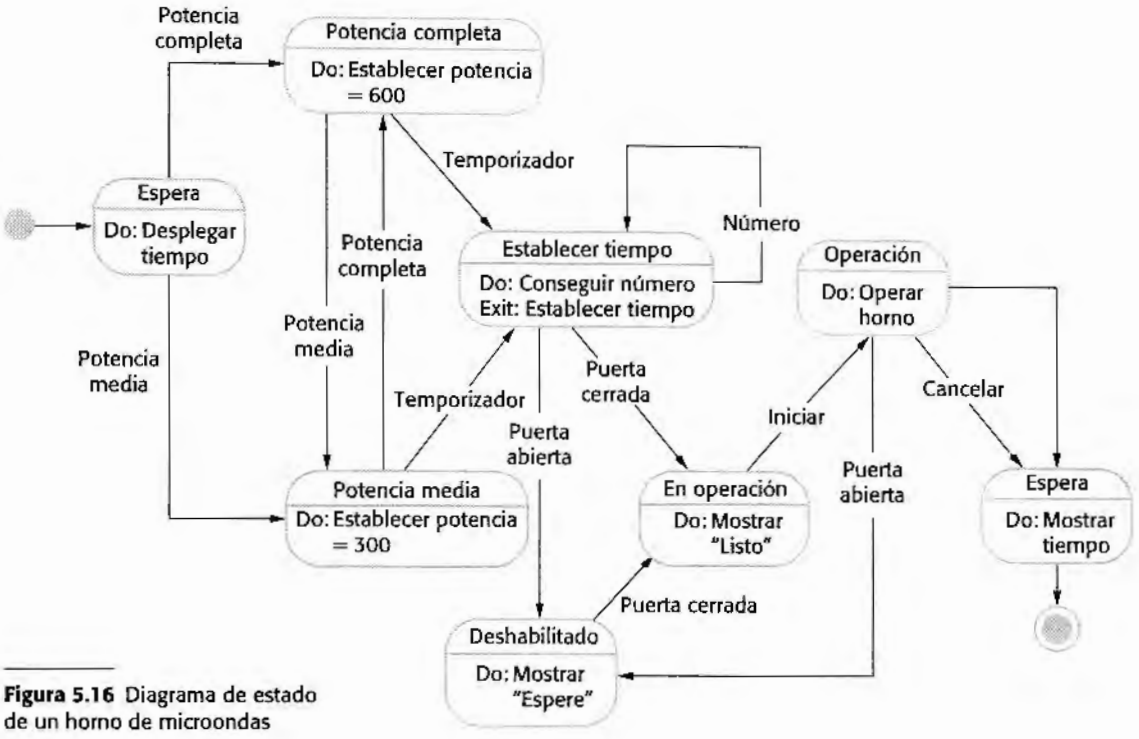


Figura 5.16 Diagrama de estado de un horno de microondas

flujo de datos dentro del sistema, pero suelen incluir información adicional acerca de los cálculos realizados en cada estado.

Se usa un ejemplo de software de control para un horno de microondas rnny sencillo, que ilustra el modelado dirigido por un evento. Los hornos de microondas reales son mucho más complejos qne este sistema, pero el sistema simplificado es más fácil de entender. Este microondas sencillo tiene un interruptor para seleccionar potencia completa o media, un teclado numérico para ingresar el tiempo de cocción, un botón de iniciar/detener y nna pantalla alfanumérica.

Se supone que la secnencia de acciones al nsar el horno de microondas es:

1. Seleccionar el nivel de potencia (ya sea media o completa)
2. Ingresar el tiempo de cocción con el teclado numérico.
3. Presionar "Iniciar", y la comida se cocina durante el tiempo dado.

Por razones de seguridad, el horno no opera cuando la pnerta esté abierta y, al completar la cocción, se escuchará un timbre. El horno tiene una pantalla alfanumérica muy sencilla que se usa para mostrar varios avisos de alerta y mensajes de advertencia.

En los diagramas de estado UML, los rectángulos redondeados representan estados del sistema. Pueden incluir una breve descripción (después de "do") de las acciones que se tomarán en dicho estado. Las flechas etiquetadas representan estímulos que fuerzan una transición de un estado a otro. Puede indicar los estados inicial y final usando círculos rellenos, como en los diagramas de actividad.

A partir de la figura 5.16 se observa que el sistema empieza en un estado de espera e, inicialmente, responde al botón de potencia completa o al botón de potencia media. Los

Estado	Descripción
Esperar	El horno espera la entrada. La pantalla indica el tiempo actual.
Potencia media	La potencia del horno se establece en 300 watts. La pantalla muestra "Potencia media".
Potencia completa	La potencia del horno se establece en 600 watts. La pantalla muestra "Potencia completa".
Establecer tiempo	El tiempo de cocción se establece al valor de entrada del usuario. La pantalla indica el tiempo de cocción seleccionado y se actualiza conforme se establece el tiempo.
Deshabilitado	La operación del horno se deshabilita por cuestiones de seguridad. La luz interior del horno está encendida. La pantalla indica "No está listo".
Habilitado	Se habilita la operación del horno. La luz interior del horno está apagada. La pantalla muestra "Listo para cocinar".
Operación	Horno en operación. La luz interior del horno está encendida. La pantalla muestra la cuenta descendente del temporizador. Al completar la cocción, suena el timbre durante cinco segundos. La luz del horno está encendida. La pantalla muestra "Cocción completa" mientras suena el timbre.

Estímulo	Descripción
Potencia media	El usuario oprime el botón de potencia media.
Potencia completa	El usuario oprime el botón de potencia completa.
Temporizador	El usuario oprime uno de los botones del temporizador.
Número	El usuario oprime una tecla numérica.
Puerta abierta	El interruptor de la puerta del horno no está cerrado.
Puerta cerrada	El interruptor de la puerta del horno está cerrado.
Iniciar	El usuario oprime el botón Iniciar.
Cancelar	El usuario oprime el botón Cancelar.

Figura 5.17 Estados y estímulos para el horno de microondas

usuarios pueden cambiar su opinión después de seleccionar uno de ellos y oprimir el otro botón. Se establece el tiempo y, si la puerta está cerrada, se habilita el botón Iniciar. Al presionar este botón comienza la operación del horno y tiene lugar la cocción durante el tiempo especificado. Éste es el final del ciclo de cocción y el sistema regresa al estado de espera.

La notación UML permite indicar la actividad que ocurre en un estado. En una especificación detallada del sistema, hay que proporcionar más detalle tanto de los estímulos como de los estados del sistema. Esto se ilustra en la figura 5.17, la cual señala una descripción tabular de cada estado y cómo se generan los estímulos que fuerzan transiciones de estado.

El problema con el modelado basado en el estado es que el número de posibles estados se incrementa rápidamente. Por lo tanto, para modelos de sistemas grandes, necesita ocultar

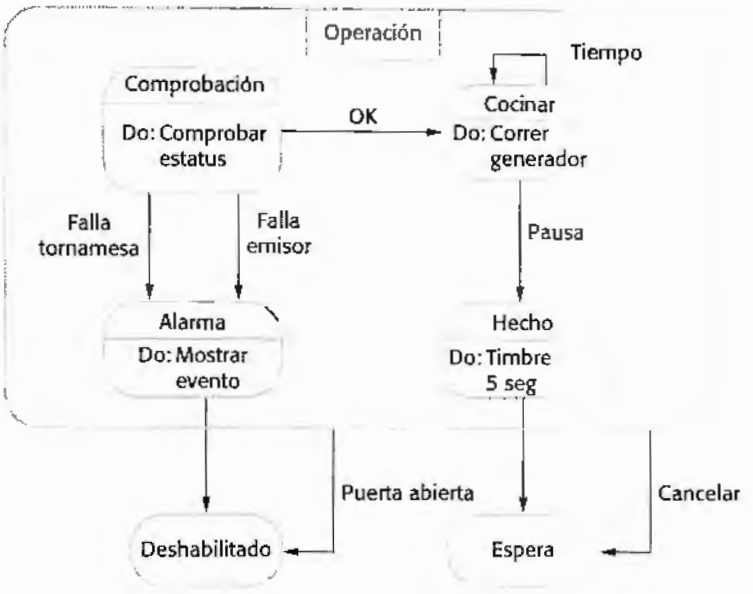


Figura 5.18 Operación del horno de microondas

detalles en los modelos. Una forma de hacer esto es mediante la noción de un superestado que encapsule algunos estados separados. Este superestado se parece a un solo estado en un modelo de nivel superior, pero entonces se expande para mostrar más detalles en un diagrama separado. Para ilustrar este concepto, considere el estado Operación en la figura 5.15. Éste es un superestado qne puede expandirse, como se ilustra en la figura 5.18.

El estado Operación incluye algunos subestados. Muestra que la operación comienza con una comprobación de estatus y que, si se descubren problemas, se indica una alarma y la operación se deshabilita. La cocción implica operar el generador de microondas durante el tiempo especificado; al terminar, snena un timbre. Si la puerta está abierta durante la operación, el sistema se mueve hacia el estado deshabilitado, como se muestra en la figura 5.15.

5.5 Ingeniería dirigida por modelo

La ingeniería dirigida por modelo (MDE, por las siglas de *Model-Driven Engineering*) es un enfoque al desarrollo de software donde los modelos, y no los programas, son las salidas principales del proceso de desarrollo (Kent, 2002; Schmidt, 2006). Los programas que se ejecutan en una plataforma hardware/software se generan en tal caso automáticamente a partir de los modelos. Los partidarios de la MDE argumentan que ésta eleva el nivel de abstracción en la ingeniería de software, pues los ingenieros ya no tienen que preocuparse por detalles del lenguaje de programación o las especificidades de las plataformas de ejecución.

La ingeniería dirigida por modelo tiene sus raíces en la arquitectura dirigida por modelo (MDA, por las siglas de *Model-Driven Architecture*), que fue propuesta por el Object Management Gronp (OMG) en 2001 como un nuevo paradigma de desarrollo de software. La ingeniería dirigida por modelo y la arqnitectura dirigida por modelo se

ven normalmente iguales. Sin embargo, se considera que la MDE tiene un ámbito más amplio que la MDA. Como se estudia más adelante en esta sección, la MDA se enfoca en las etapas de diseño e implementación del desarrollo de software, mientras que la MDE se interesa por todos los aspectos del proceso de ingeniería de software. Por lo tanto, los temas como ingeniería de requerimientos basada en un modelo, procesos de software para desarrollo basado en un modelo, y pruebas basadas en un modelo son parte de MDE, pero no, en este momento, de la MDA.

Aunque la MDA se usa desde 2001, la ingeniería basada en modelo aún está en una etapa temprana de desarrollo, y no es claro si tendrá o no un efecto significativo sobre la práctica de ingeniería de software. Los principales argumentos a favor y en contra de MDE son:

1. *En favor de la MDE* La ingeniería basada en modelo permite a los ingenieros pensar sobre sistemas en un nivel de abstracción elevado, sin ocuparse por los detalles de su implementación. Esto reduce la probabilidad de errores, acelera el diseño y el proceso de implementación, y permite la creación de modelos de aplicación reutilizables, independientes de la plataforma de aplicación. Al usar herramientas poderosas, las implementaciones de sistema pueden generarse para diferentes plataformas a partir del mismo modelo. En consecuencia, para adaptar el sistema a alguna nueva plataforma tecnológica, sólo es necesario escribir un traductor para dicha plataforma. Cuando está disponible, todos los modelos independientes de plataforma pueden reubicarse rápidamente en la nueva plataforma.

2. *Contra la MDE* Como se analizó anteriormente en este capítulo, los modelos son una buena forma de facilitar las discusiones sobre un diseño de software. Sin embargo, no siempre se sigue que las abstracciones que soporta el modelo son las abstracciones correctas para la implementación. De este modo, es posible crear modelos de diseño informal, pero siendo así, el sistema se implementa usando un paquete configurable comercial (*off-the-shelf*). Más aún, los argumentos para independencia de plataforma sólo son válidos para sistemas grandes de larga duración, donde las plataformas se vuelven obsoletas durante el tiempo de vida de un sistema. Sin embargo, para esta clase de sistemas, se sabe que la implementación no es el principal problema: ingeniería de requerimientos, seguridad y confiabilidad, integración con sistemas heredados, y las pruebas son más significativos.

El OMG reporta en sus páginas Web (www.omg.org/mda/products_success.htm) historias de éxito reveladoras de la MDE y el enfoque utilizado dentro de grandes compañías como IBM y Siemens. Las técnicas se usaron con éxito en el desarrollo de grandes sistemas de software de larga duración, como sistemas de manejo de tráfico aéreo. No obstante, en el momento de escribir este texto, los enfoques dirigidos por modelo no son ampliamente usados por la ingeniería de software. Como los métodos formales de la ingeniería de software, que se estudian en el capítulo 12, se considera que MDE es un importante desarrollo. Pero, como también es el caso con los métodos formales, no es claro si los costos y riesgos de los enfoques dirigidos por modelo superan los posibles beneficios.

5.5.1 Arquitectura dirigida por modelo

La arquitectura dirigida por modelo (Kleppe *et al.*, 2003; Mellor *et al.*, 2004; Stahl y Voelter, 2006) es un enfoque orientado a un modelos para el diseño y la implementación de software, que usa un subconjunto de modelos UML para describir un sistema. Aquí, se

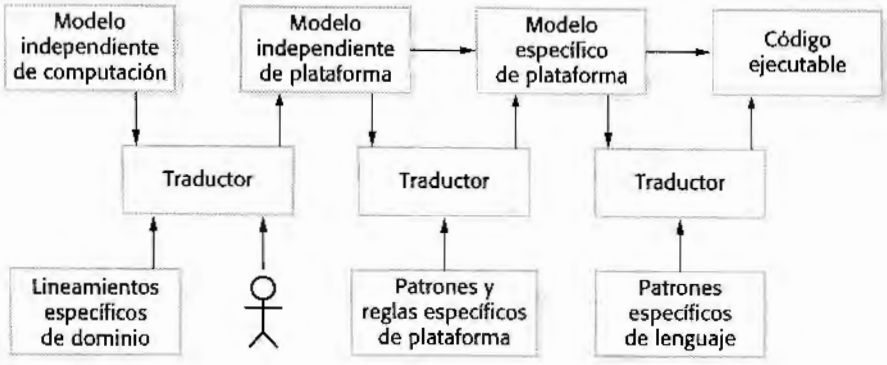


Figura 5.19 Transformaciones MDA

crean modelos a diferentes niveles de abstracción. A partir de un modelo independiente de plataforma de alto nivel, es posible, en principio, generar un programa funcional sin intervención manual.

El método de MDA recomienda la producción de tres tipos de modelo de sistema abstracto:

1. Un modelo independiente de computación (CIM) qne modela las importantes abstracciones de dominio nsadas en el sistema. En ocasiones, los CIM se llaman modelos de dominio. Es posible desarrollar varios CIM diferentes, que reflejen distintas percepciones del sistema. Por ejemplo, puede haber un CIM de seguridad, en el cual se identifiquen abstracciones de seguridad importantes, como un CIM de activo, un rol y nn registro del paciente, que describan abstracciones como pacientes, consultas, etcétera.
2. Un modelo independiente de plataforma (PIM) que modele la operación del sistema sin referencia a su implementación. El PIM se describe usnalmente mediante modelos UML qne muestran la estructura estática del sistema y cómo responde a eventos externos e internos.
3. Modelos específicos de plataforma (PSM) qne son transformaciones del modelo independiente de plataforma con un PSM separado para cada plataforma de aplicación. En principio, puede haber capas de PSM, y cada una agrega cierto detalle específico de la plataforma. De este modo, el PSM de primer nivel podría ser específico de "middleware", pero independiente de la base de datos. Cuando se elige una base de datos específica, podría generarse entonces un PSM específico de base de datos.

Como vimos, las transformaciones entre dichos modelos pueden definirse y aplicarse antomáticamente con herramientas de software. Esto se ilustra en la figura 5.19, qne también muestra un nivel final de transformación automática. Una transformación se aplica al PSM para generar un código ejecntable que opere en la plataforma de software designada.

En el momento de escribir este texto, la tradncción automática CIM a PIM todavía está en etapa de investigación de prototipo. Es improbable que en el fnturo cercano estén disponibles herramientas de traducción completamente automatizadas. Para el futuro

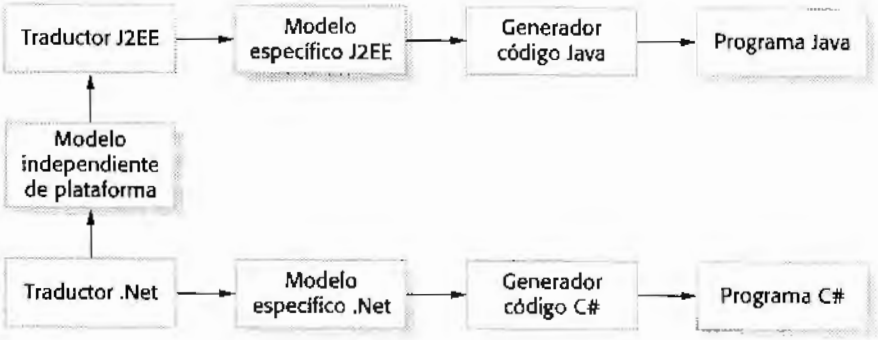


Figura 5.20 Múltiples modelos específicos de plataforma

previsible se necesitará la intervención humana, lo que se indica mediante una figura ilustrativa en la figura 5.19. Los CIM se relacionan, y parte del proceso de traducción puede involucrar conceptos de vinculación en diferentes CIM. Por ejemplo, el concepto de un papel en un CIM de seguridad que puede trazarse dentro del concepto de un miembro de personal en un CIM de hospital. Mellor y Balcer (2002) dan el nombre de "puentes" a la información que soporta el mapeo de un CIM a otro.

La traducción de PIM a PSM es más madura y se dispone de varias herramientas comerciales que proporcionan traductores de PIM a plataformas comunes como Java y J2EE. Éstas se apoyan en una extensa librería de reglas y patrones específicos de plataforma para convertir el PIM al PSM. Puede haber muchos PSM para cada PIM en el sistema. Si se tiene la intención de que un sistema de software funcione en diferentes plataformas (por ejemplo J2EE y .NET), entonces sólo es necesario mantener el PIM. Los PSM para cada plataforma se generan automáticamente. Esto se ilustra en la figura 5.20.

Aunqne las herramientas de soporte MDA incluyen traductores específicos de plataforma, es frecuente el caso de que sólo ofrezcan soporte parcial para la traducción de PIM a PSM. En la gran mayoría de los casos, el entorno de ejecución para un sistema es más qne la plataforma de ejecución estándar (por ejemplo, J2EE, .NET, etcétera). También incluye otros sistemas de aplicación, librerías de aplicación que son específicas a una compañía y librerías de interfaz de usnario. Como éstas varían significativamente de una compañía a otra, no está disponible un soporte estándar para herramientas. Por lo tanto, cuando se introduce MDA, quizá deban crearse traductores de propósito especial que tomen en cnenta las características del entorno local. En algunos casos (por ejemplo, para generación de interfaz de usuario), la traducción de PIM a PSM completamente antomatizada es imposible.

Existe una relación difícil entre métodos ágiles y arquitectura dirigida por modelo. La noción de modelado frontal extenso contradice las ideas fundamentales del manifiesto ágil y se conjetura que pocos desarrolladores ágiles se sienten cómodos con la ingeniería dirigida por modelo. Los desarrolladores de MAD afirman qne se tiene la intención de apoyar un enfoque iterativo para el desarrollo y, por lo tanto, puede usarse dentro de los métodos ágiles (Mellor *et al.*, 2004). Si las transformaciones pueden automatizarse completamente y a partir de un PIM se genera un programa completo, entonces, en principio, MDA podría usarse en un proceso de desarrollo ágil, ya que no se requeriría codificación separada. Sin embargo, hasta donde se sabe, no hay herramientas de MDA que soporten prácticas como las pruebas de regresión y el desarrollo dirigido por pruebas.