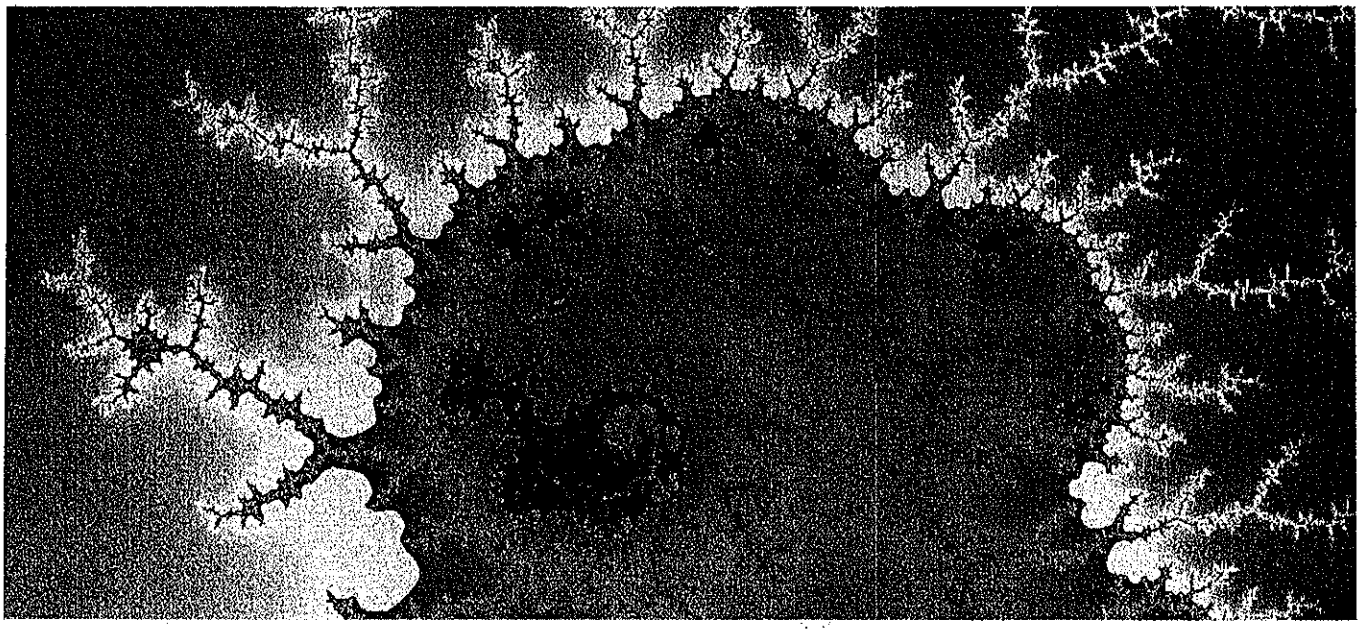
Parte 5

Implementación



23

El workflow de implementación

23.1. Presentación del capítulo

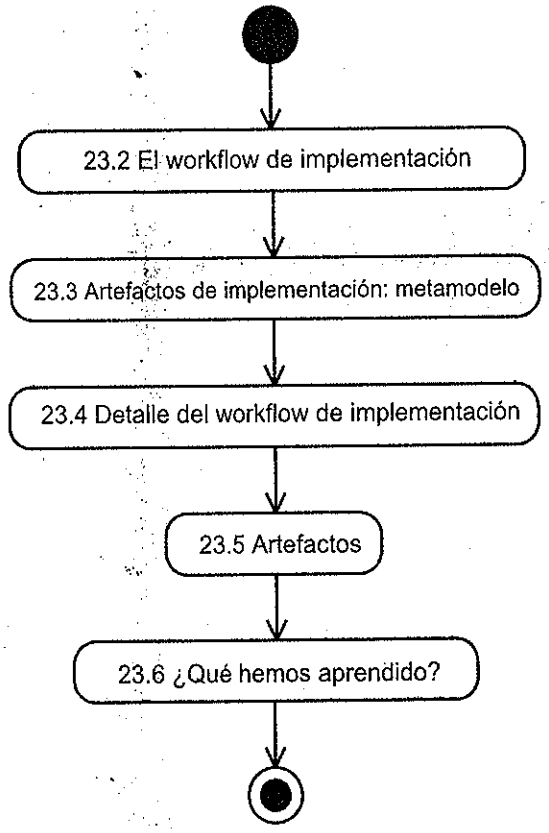


Figura 23.1.

Existe muy poco trabajo para el analista/diseñador orientado a objetos en el workflow de implementación, por lo que ésta es la parte más breve del libro. Sin embargo, la implementación implica cierto examen ya que, aunque la actividad principal en el workflow de implementación es generar código, verá que siguen estando implicados algunos elementos del modelado UML.

23.2. El workflow de implementación

El workflow de implementación empieza en serio en la fase de elaboración y es el foco principal de la fase de construcción (véase la figura 23.2).

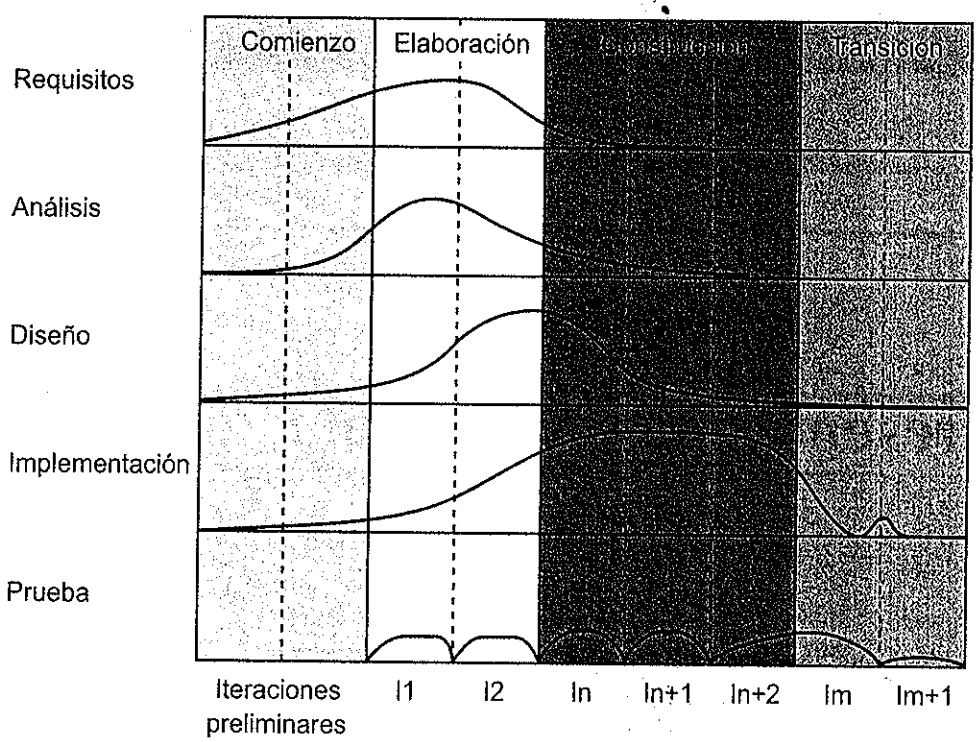


Figura 23.2. Adaptada de la figura 1.5 [Jacobson 1] con permiso de Addison-Wesley.

La implementación trata sobre transformar un modelo de diseño en código ejecutable. Desde el punto de vista del analista/diseñador, la finalidad de la implementación es generar un modelo de implementación si eso es necesario. Este modelo implica la asignación de clases de diseño a componentes. Cómo se realice esto depende en gran medida del lenguaje de programación destino.

El foco principal del workflow de implementación es generar código ejecutable. La producción de un modelo de implementación puede ser un subproducto de este foco, en lugar de una actividad explícita de modelado. De hecho, muchas herramientas de modelado le permiten invertir-generar un modelo de implementación desde el código fuente. Esta estrategia deja el modelado de la implementación a los programadores. Sin embargo, existen dos casos en los que una actividad explícita de modelado de implementación, realizada por analistas/diseñadores orientados a objetos, podría ser muy importante.

- Si trata de generar código directamente desde el modelo, necesitará especificar detalles como archivos origen y componentes (a menos que tome los elementos predeterminados de la herramienta de modelado).
- Si está realizando desarrollo basado en componentes para reutilizar componentes, la asignación de clases de diseño e interfaces a componentes se convierte en un aspecto estratégico. Querrá modelar esto primero en lugar de dejarlo a los programadores.

En este capítulo examinamos lo que implica crear un modelo de implementación.

23.3. Artefactos de implementación: metamodelo

La relación entre el modelo de implementación y el modelo de diseño es muy sencilla. El modelo de implementación es la vista de implementación de un modelo de diseño, es decir, es parte del modelo de diseño. Esto se muestra en la figura 23.3.

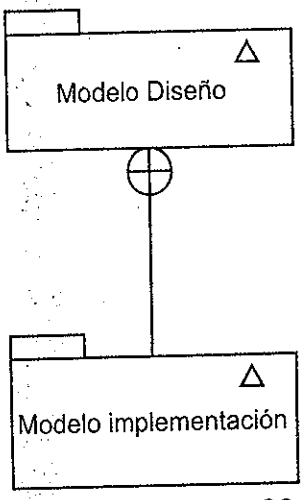


Figura 23.3.

El modelo de implementación es la parte del modelo de diseño que trata con aspectos de implementación. Especifica cómo los elementos de diseño se muestran por artefactos y cómo estos artefactos se despliegan en nodos. Los artefactos representan las especificaciones de los elementos del mundo real como archivos fuente y los nodos representan las especificaciones de hardware o entornos de ejecución en las que se despliegan estos elementos. La relación entre el modelo de diseño y el modelo de implementación se ilustra en la figura 23.4.

La relación `<<manifest>>` entre los artefactos y los componentes indica que los artefactos son las representaciones físicas de los componentes. Por ejemplo, un componente puede englobar una clase y una interfaz, y ambos se realizan por un solo artefacto, un archivo que contiene código fuente.

Los componentes de diseño son entidades lógicas que agrupan elementos de diseño, pero los artefactos de implementación mapean con mecanismos de agrupación reales del lenguaje de implementación destino.

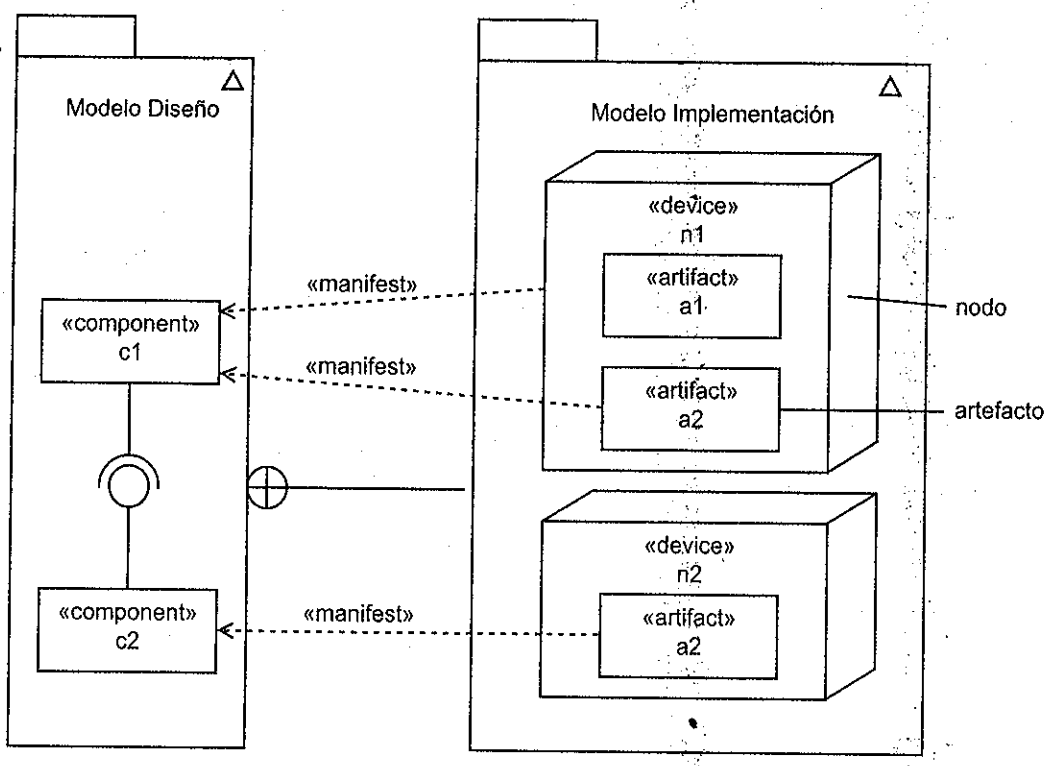


Figura 23.4.

23.4. Detalle del workflow de implementación

Como puede ver por la figura 23.5, el workflow de implementación implica al arquitecto, integrador del sistema e ingeniero de componentes. Analistas/diseñadores individuales o pequeños grupos de analistas/diseñadores pueden desempeñar cualquiera de estos roles en el workflow de implementación. Su foco estará en generar los modelos de despliegue e implementación (parte de la implementación de arquitectura). La integración del sistema, la implementación de clase y las pruebas están más allá del alcance de este libro; éstas son actividades de programación en lugar de actividades de análisis y diseño. (Observe que en la figura 23.5 hemos actualizado la figura original desde `Implementar un subsistema` hasta el más general `Implementar un componente` ya que esto es más correcto desde la perspectiva de UML 2.)

23.5. Artefactos

El artefacto clave del workflow de implementación desde el punto de vista del analista/diseñador orientado a objetos es el modelo de implementación. Este modelo consta de diagramas de componentes para mostrar cómo los artefactos muestran componentes y un nuevo tipo de diagrama, el diagrama de despliegue. El diagrama de despliegue modela los nodos computacionales físicos en los que se desplegarán los artefactos de software y las relaciones entre esos nodos. Examinamos los diagramas de despliegue en detalle en el capítulo 24.

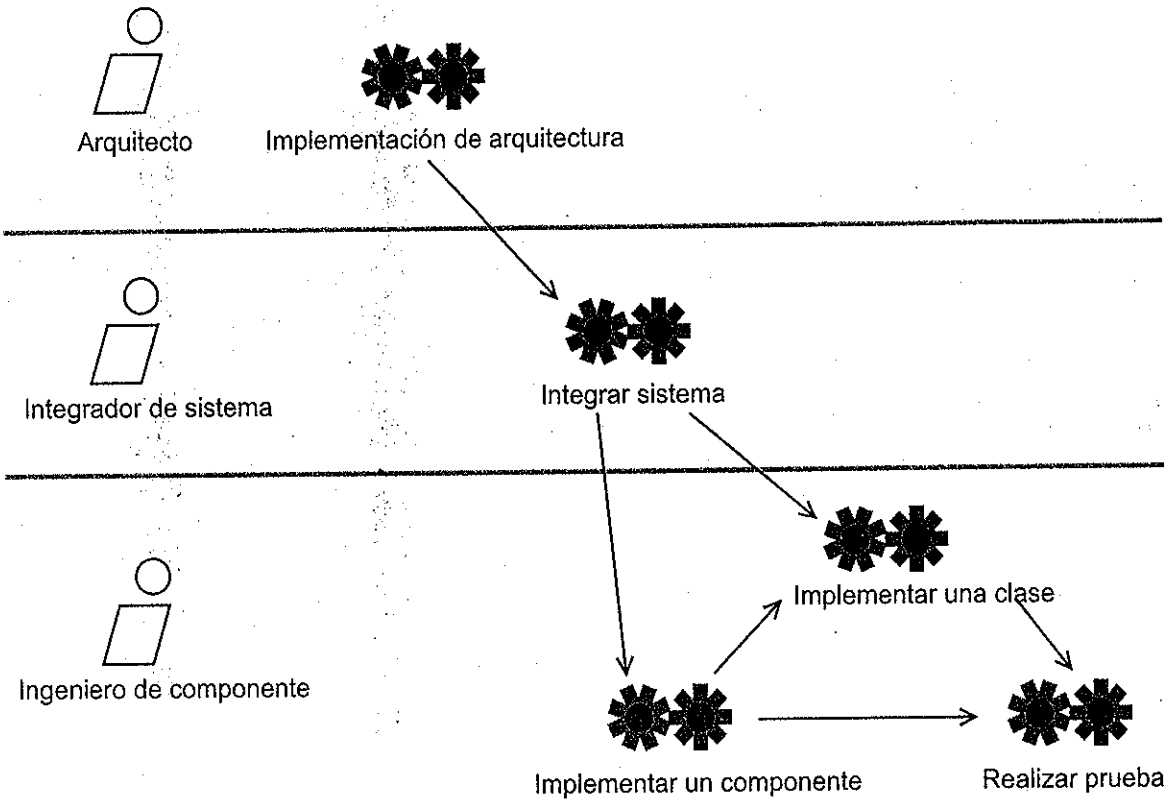
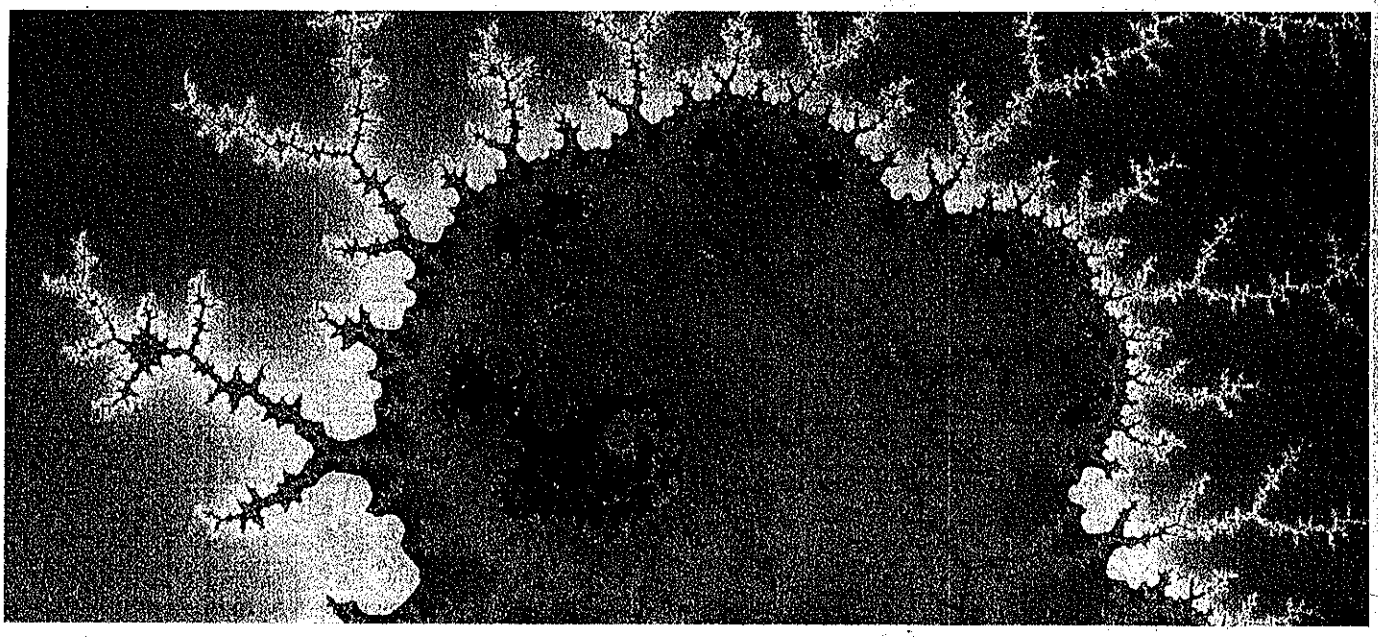


Figura 23.5. Adaptada de la figura 10.16 [Jacobson 1] con permiso de Addison-Wesley.

23.6. ¿Qué hemos aprendido?

Implementar trata sobre crear código. Sin embargo, al analista/diseñador orientado a objetos se le puede pedir que cree un modelo de implementación. Ha aprendido lo siguiente:

- El workflow de implementación es el foco principal de la fase de construcción.
- Implementación trata sobre la transformación de un modelo de diseño en código ejecutable.
- El modelado de implementación es importante cuando:
  - Trata de generar código desde el modelo.
  - Realiza desarrollo basado en componentes para reutilizar componentes.
- El modelo de implementación es parte del modelo de diseño.
- Los artefactos representan las especificaciones de elementos del mundo real como archivos fuente:
  - Los componentes se muestran por artefactos.
  - Los artefactos se despliegan en nodos.
- Los nodos representan las especificaciones de hardware o entornos de ejecución.



24
Despliegue

24.1. Presentación del capítulo

En este capítulo examinamos la actividad UP `Implementación de arquitectura` y la forma de generar un diagrama de despliegue. Éste es un diagrama que muestra cómo el software que está desarrollando se desplegará sobre hardware físico y cómo se conecta ese hardware. Presentamos un sencillo ejemplo Java.

24.2. Actividad UP: Implementación de arquitectura

Esta actividad trata sobre identificar componentes significativos arquitectónicamente y mapearlos con hardware físico; trata sobre modelar la estructura física y distribución del sistema. La frase clave es "significativo arquitectónicamente". En principio, podría modelar el despliegue físico del sistema de forma exhaustiva. En la práctica, esto probablemente añadiría poco valor ya que los detalles exactos del despliegue de muchos componentes tendrán poca importancia arquitectónica. La excepción para esto es si genera código desde el modelo. En este caso, podría necesitar un modelo de despliegue mucho más detallado para que su generador sepa dónde situar sus artefactos de salida y pueda crear los descriptores apropiados de despliegue y archivos de creación.

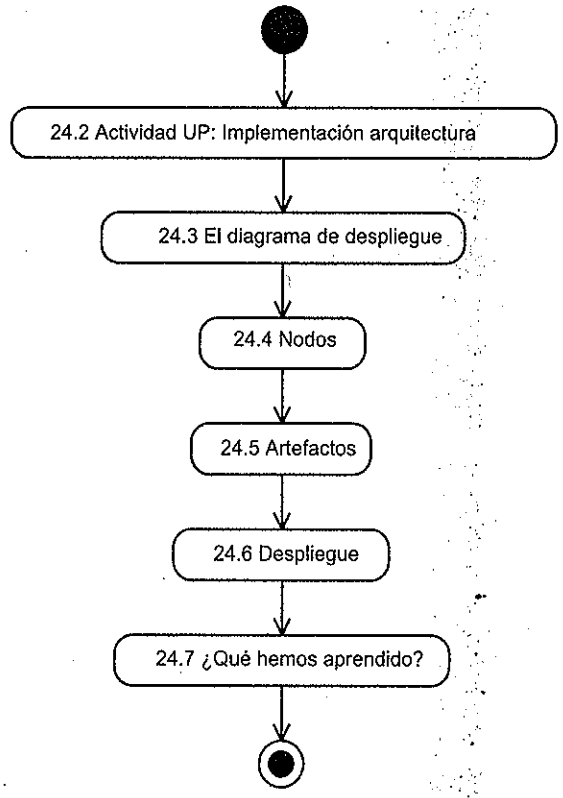


Figura 24.1.

La actividad UP `Implementación de arquitectura` se muestra en la figura 24.2. Hemos modificado esta figura del original de dos formas y hemos identificado los cambios al poner en gris los artefactos afectados.

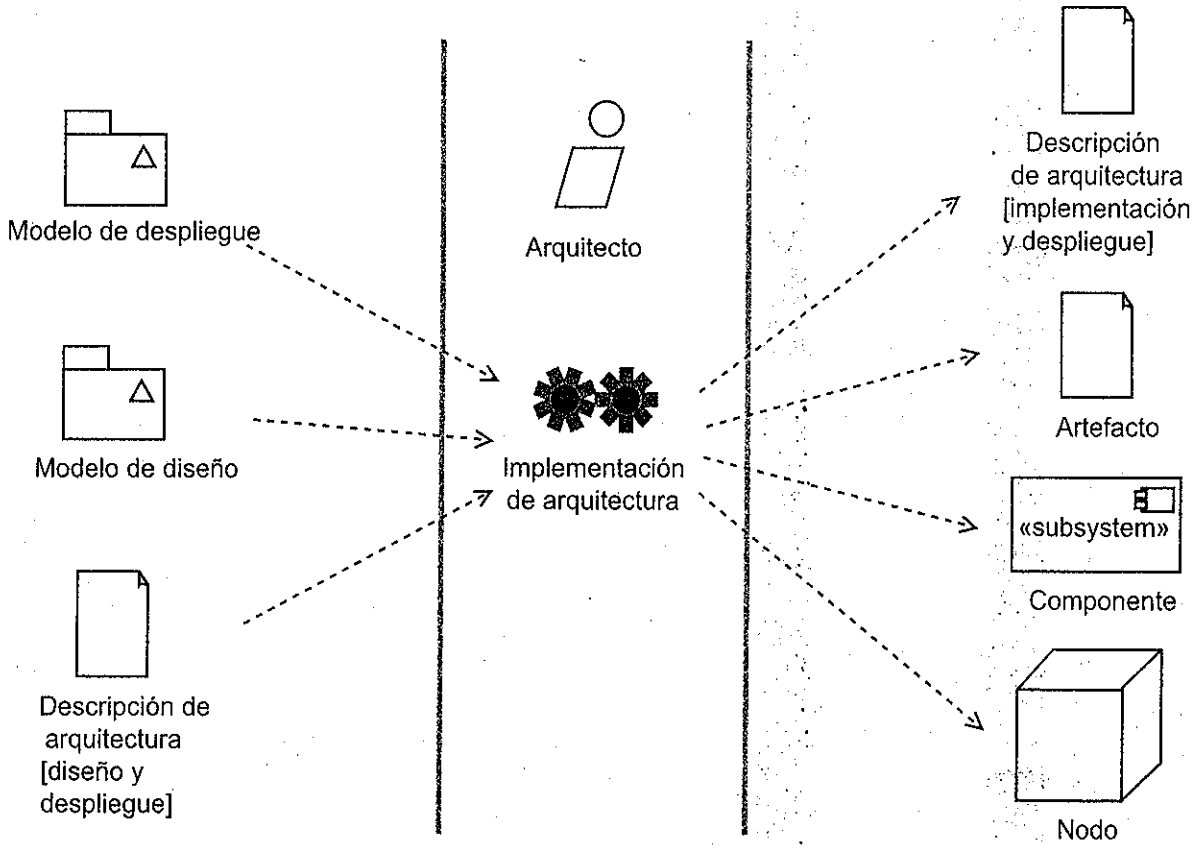


Figura 24.2. Adaptada de la figura 10.17 [Jacobson 1] con permiso de Addison-Wesley.

- Según UML 2 mostramos un subsistema como un componente estereotipado en lugar de como un paquete estereotipado.
- Hemos mostrado explícitamente los artefactos y nodos que salen de la actividad. Éstos estaban implícitos en la figura original.

Desde el punto de vista del analista/diseñador orientado a objetos, la actividad clave en `Implementar arquitectura` es crear uno o más diagramas de despliegue. El diagrama de despliegue sitúa juntos componentes, artefactos y nodos para especificar la arquitectura física del sistema. Examinamos este diagrama en detalle en el resto del capítulo.

La otra actividad es actualizar la `descripción de arquitectura` con detalles de despliegue e implementación importantes arquitectónicamente.

24.3. El diagrama de despliegue

En UML el despliegue es el proceso de asignar artefactos a nodos o instancias de artefactos a instancias de nodos. Examinamos artefactos y nodos en detalle en breve.

El diagrama de despliegue especifica el hardware físico sobre el que el sistema de software se ejecutará y también especifica cómo el software se despliega en ese hardware.

El diagrama de despliegue mapea la arquitectura de software creada en diseño con una arquitectura física de sistema que lo ejecuta. En sistemas distribuidos, modela la distribución del software a través de nodos físicos.

Existen dos formas de diagrama de despliegue:

1. **Forma de descriptor:** Contiene nodos, relaciones entre nodos y artefactos. Un nodo representa un tipo de hardware (como un PC). De forma similar, un artefacto representa un tipo de artefacto de software físico como un archivo JAR de Java.
2. **Forma de instancia:** Contiene instancias de nodo, relaciones entre instancias de nodo e instancias de artefacto. Una instancia de nodo representa una pieza de hardware identificable y específica (como el PC de Jim). Una instancia de artefacto representa una instancia específica de un tipo de software, como la copia particular de FrameMaker (`www.adobe.com`) utilizada para escribir esto o un archivo JAR determinado. Si no conoce los detalles de instancias específicas, puede utilizar instancias anónimas.

Aunque lo trataremos como una actividad de implementación, un diagrama de despliegue se crea a menudo en diseño como parte del proceso de decidir la arquitectura final de hardware. Podría empezar por crear un diagrama de despliegue en forma de descriptor limitado a nodos y las conexiones entre ellos. Luego puede mejorar esto en uno o más diagramas de despliegue en forma de instancia mostran-

do posibles agrupaciones de instancias anónimas de nodo. Cuando sabe los detalles del hardware en un sitio de despliegue, puede crear un diagrama de despliegue en forma de instancia mostrando las máquinas reales en ese sitio, si fuera necesario.

La construcción de un diagrama de despliegue es por lo tanto un proceso en dos pasos:

1. En el workflow de diseño, céntrese principalmente en el nodo o instancias de nodo y conexiones.
2. En el workflow de implementación, céntrese en asignar instancias de artefactos a instancias de nodo (forma de instancia) o artefactos a nodos (forma de descriptor).

En los siguientes apartados examinamos los nodos y artefactos en detalle.

24.4. Nodos

La especificación UML 2.0 [UML2S] indica: "un nodo representa un tipo de recurso computacional sobre el que se pueden desplegar los artefactos para su ejecución". Existen dos estereotipos estándar para nodos:

- `<<device>>`: El nodo representa un tipo de dispositivo físico como un PC o un servidor Sun Fire.
- `<<execution environment>>`: El nodo representa un tipo de entorno de ejecución para software como un servidor Web Apache o el contenedor EJB (Enterprise JavaBeans) JBoss.

Los nodos se pueden anidar en nodos. Por ejemplo, el diagrama de despliegue de forma de descriptor muestra que cero o más `PCWindows` que ejecutan el navegador Web `Firefox` se pueden conectar a cero o más servidores Web `Apache` cada uno de ellos ejecutándose sobre un `PCLinux`. Observe que al nombrar los nodos `PCWindows` y `PCLinux` hemos incluido el tipo de hardware (PC) y el sistema operativo, el entorno de ejecución para todo el software que se ejecuta en esos dispositivos. Ésta es una práctica común ya que tener un nodo aparte de entorno de ejecución específicamente para el sistema operativo satura el diagrama. Mostramos `Firefox` como un entorno de ejecución porque puede ejecutar componentes de plugin como apples Java.

Una asociación entre nodos representa un canal de comunicación sobre el que se puede pasar la información. En la figura 24.3 hemos estereotipado la asociación `<<http>>` para indicar que representa una conexión HTTP (*HyperText Transport Protocol*, o Protocolo de Transporte Hipertexto) entre los dos nodos.

Si quiere mostrar instancias específicas de nodos, puede utilizar el diagrama de despliegue de forma de instancia ilustrado en la figura 24.4. La figura muestra dos PC, `PCJim` y `PCIlas` conectados a la máquina Linux `ServidorWeb1`. En la forma de instancia, las instancias de nodo representan dispositivos físicos reales o instan-

cias de entornos de ejecución que se ejecutan sobre esos dispositivos. Subrayamos los nombres de elemento para indicar que representan instancias de nodo.

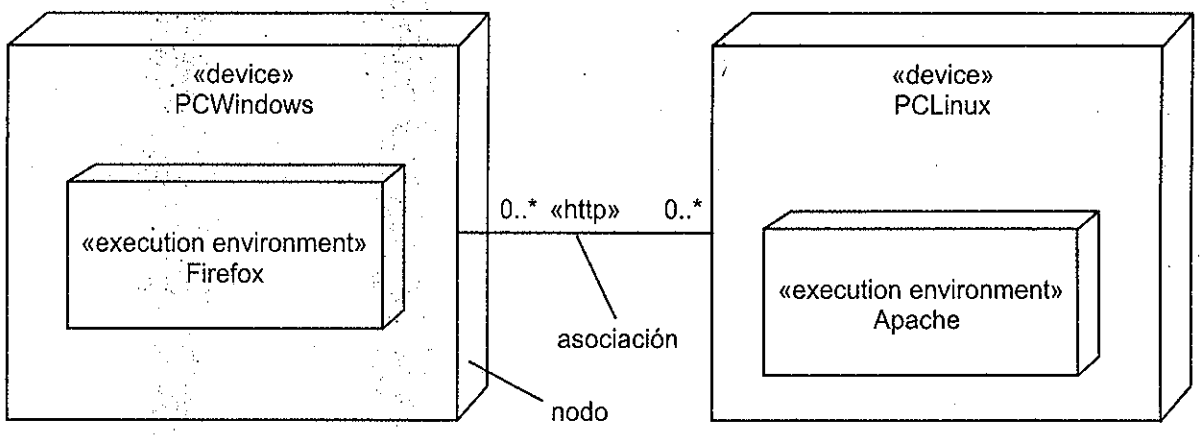


Figura 24.3.

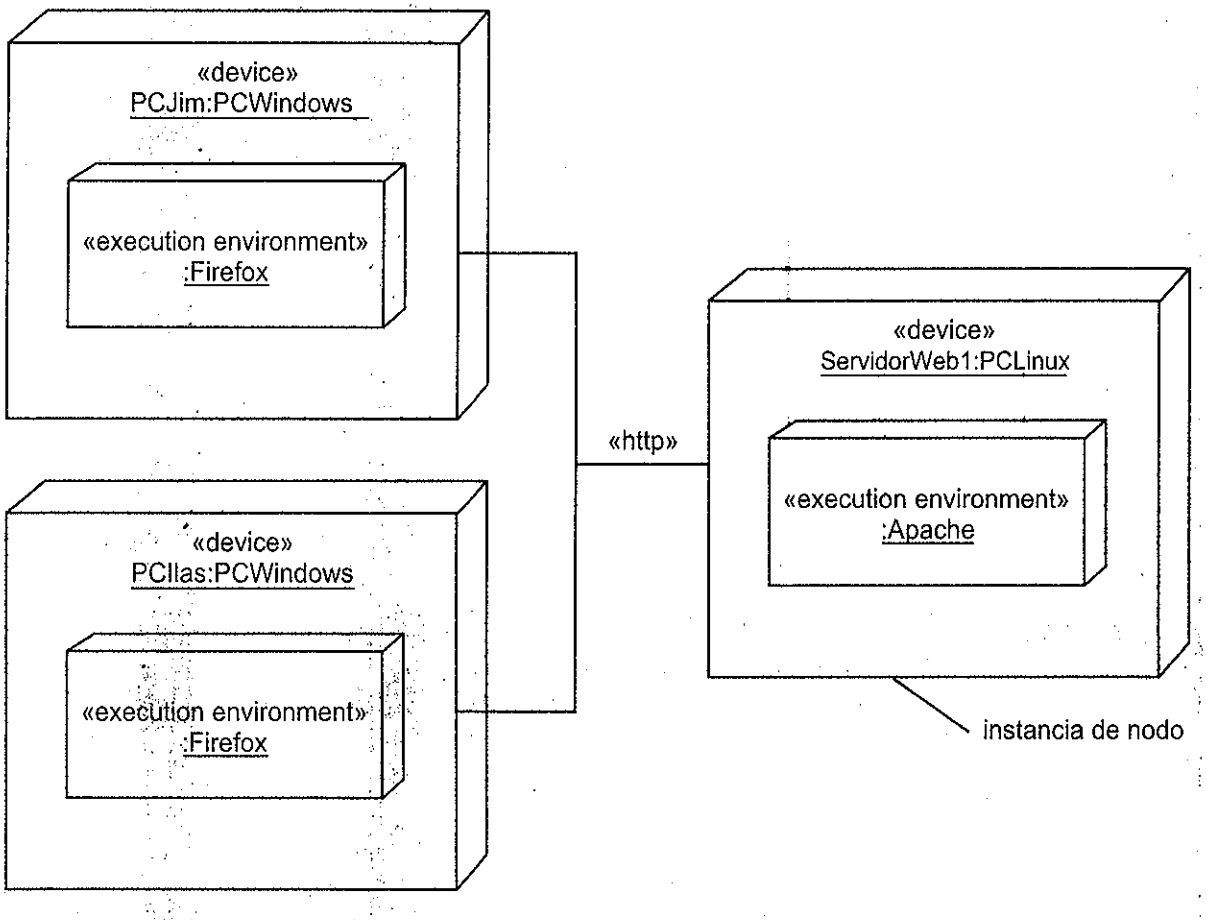


Figura 24.4.

Los diagramas de despliegue en forma de descriptor son buenos para modelar un tipo de arquitectura física. Los diagramas de despliegue en forma de instancia son buenos para modelar un despliegue real de esa arquitectura en un sitio determinado.

Según el *The UML User Guide* [Booch 2], los diagramas de despliegue son la parte más estereotipada de UML. Puede asignar sus propios iconos a estereotipos, y esto

le permite utilizar símbolos en el diagrama de despliegue que se parece tanto como sea posible al hardware actual. Esto facilita la lectura del diagrama de despliegue. Disponer de una amplia colección de clipart nos ayudará con esto. Un ejemplo de un diagrama de despliegue en forma de descriptor totalmente estereotipado se muestra en la figura 24.5.

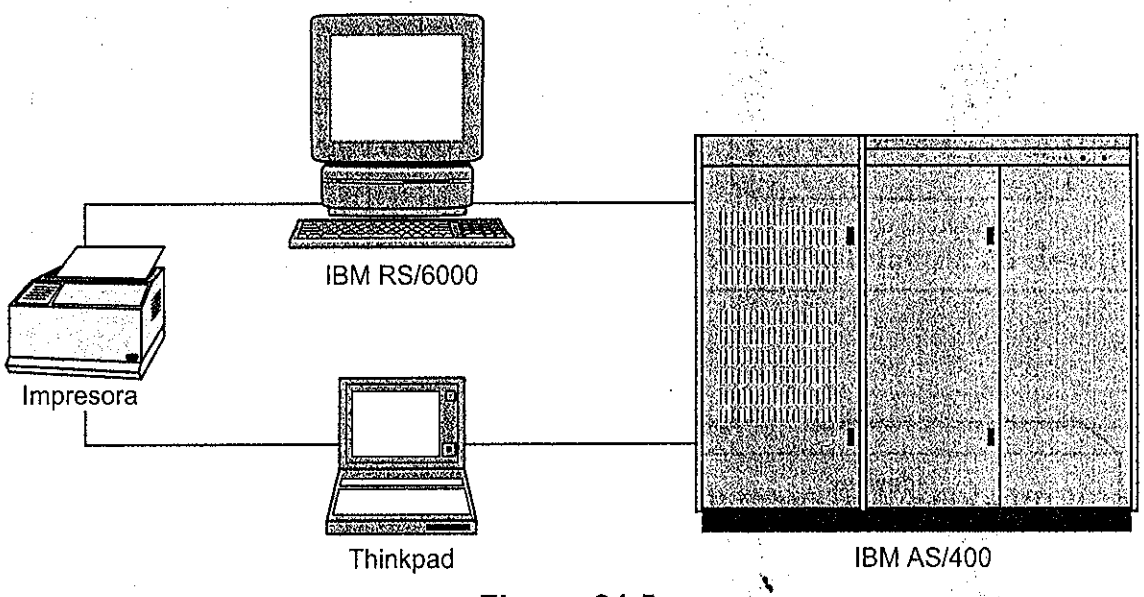


Figura 24.5.

24.5. Artefactos

Un artefacto representa la especificación de un elemento concreto del mundo real como el archivo fuente `CuentaBancaria.java`. Los artefactos se despliegan en nodos. Algunos ejemplos de artefactos son:

- Archivos fuente.
- Archivos ejecutables.
- Scripts.
- Tablas de base de datos.
- Documentos.
- Salidas del proceso de desarrollo, por ejemplo un modelo UML.

Una instancia de artefacto representa una instancia específica de un artefacto determinado, por ejemplo, una copia específica de `CuentaBancaria.java` desplegada en una máquina determinada. Las instancias de artefacto se despliegan en instancias de nodo.

Un artefacto puede proporcionar la manifestación física para cualquier tipo de elemento UML. Normalmente, muestran uno o más componentes según se ilustra en la figura 24.6. Esta figura muestra un artefacto, `sistemaBiblioteca.jar` que

muestra tres componentes, `Libro`, `Biblioteca` y `Solicitud`. Los artefactos se etiquetan con el estereotipo `<<artifact>>` y pueden tener un icono de artefacto situado en su esquina superior derecha como se muestra en la figura. La figura también muestra que los artefactos pueden depender de otros artefactos. En este caso, el artefacto `sistemaBiblioteca.jar` depende del artefacto `jdom.jar`.

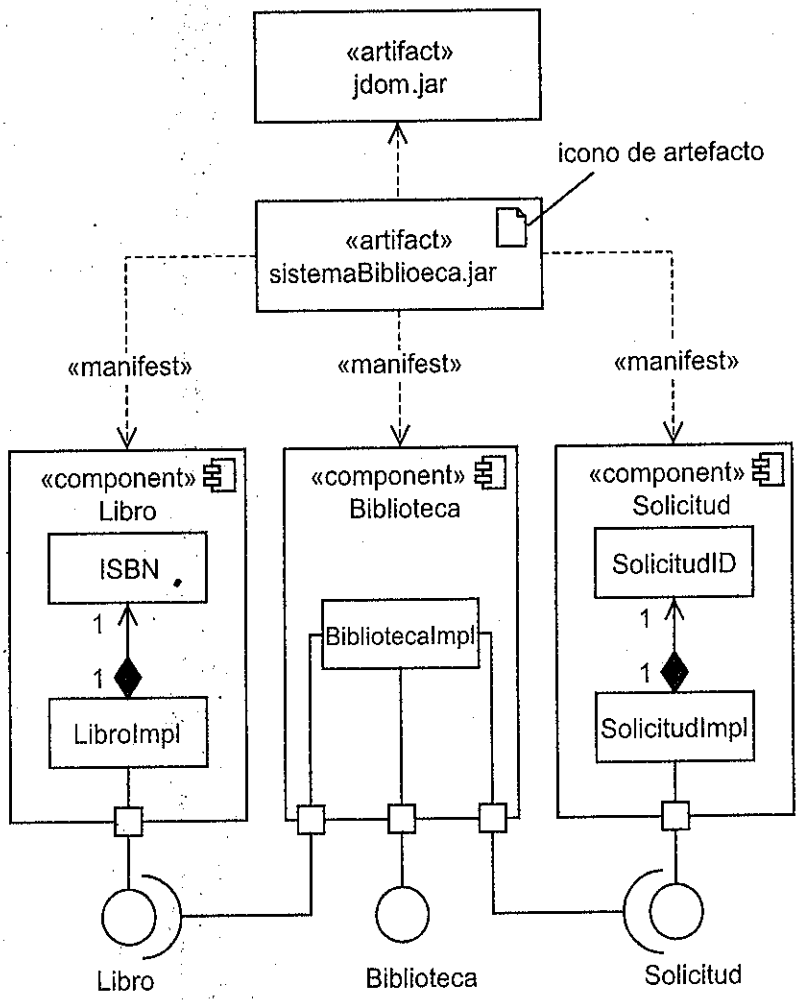


Figura 24.6.

Al igual que un nombre, todo artefacto tiene un nombre de archivo en su especificación que indica la ubicación física del artefacto. Por ejemplo, este nombrearchivo podría especificar un URL donde se encontrara la copia maestro del artefacto. Las instancias de artefacto tienen nombre de archivo que apuntan a la ubicación física de la instancia.

Examinemos el archivo JAR en la figura 24.6 en más detalle. Para crear este JAR, realiza dos pasos:

1. Compile los archivos fuente Java para las clases `Libro`, `ISBN`, `LibroImpl`, `Biblioteca`, `BibliotecaImpl`, `Solicitud`, `SolicitudID`, y `SolicitudImpl`.
2. Utilice la herramienta jar de Java para crear un archivo JAR a partir de estos archivos compilados.

Esto crea el archivo JAR que se muestra en la figura 24.7. Puede ver que este archivo JAR contiene un archivo de clase Java para cada clase e interfaz en el sistema. También contiene un directorio, META_INF, que contiene un archivo denominado MANIFEST.MF. Este archivo está generado por la herramienta jar y describe los contenidos del JAR. Observe en la figura cómo puede mostrar dependencias entre artefactos y artefactos anidados dentro de artefactos.

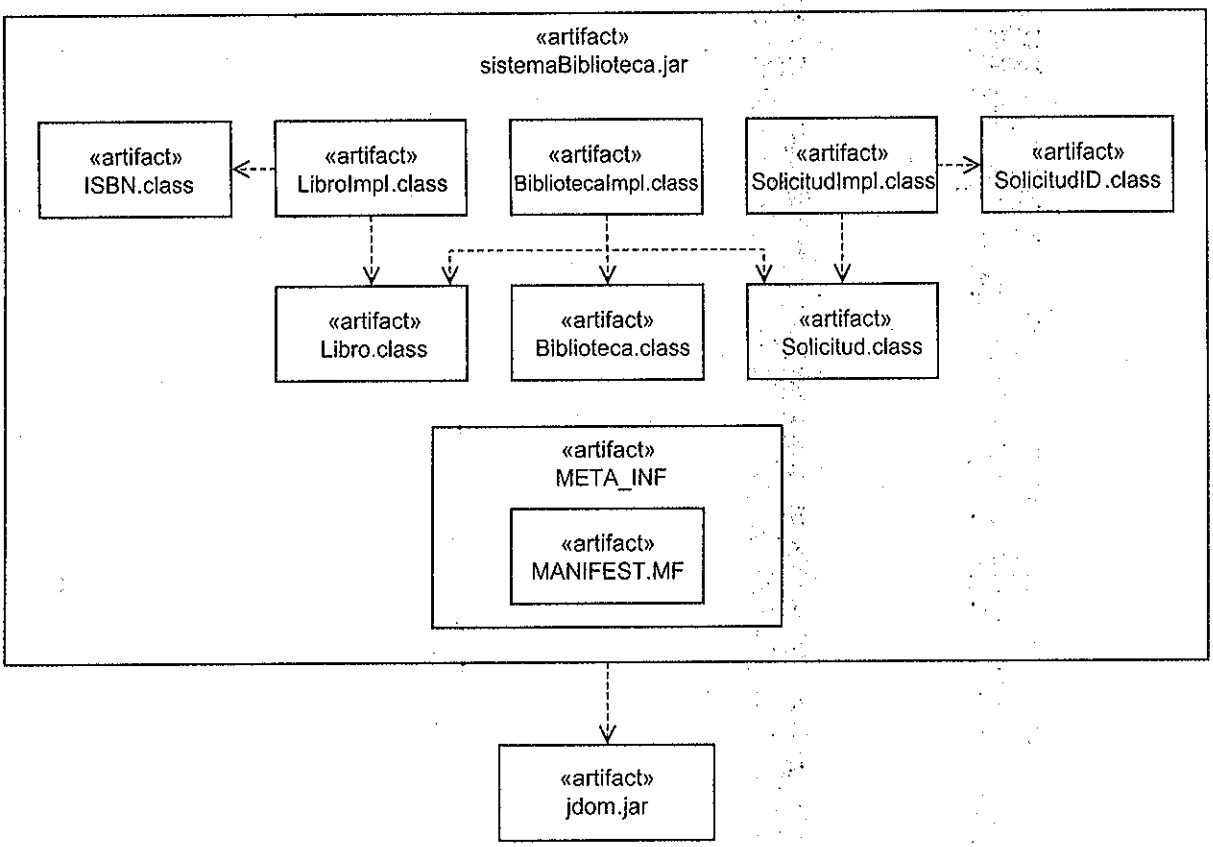


Figura 24.7.

Aunque la figura 24.7 es correcta desde una perspectiva de modelado UML, no es particularmente descriptiva ya que todo es un artefacto. La extensión `.class` le dice que alguno de los artefactos representan archivos compilados de clase Java pero no es fácil decir, por ejemplo, que META_INF representa un directorio. Esto destaca la necesidad de estereotipar artefactos para indicar claramente qué representa exactamente cada uno de ellos. UML proporciona un pequeño número de estereotipos estándar de artefactos que representan diferentes tipos de archivos. Estos se listan en la tabla 24.1.

Tabla 24.1.

Estereotipo de artefacto	Semántica
`<<file>>`	Un archivo físico.
`<<deployment spec>>`	Una especificación de detalles de despliegue (por ejemplo, web.xml en J2EE).

Estereotipo de artefacto	Semántica
<<document>>	Un archivo genérico que contiene cierta información.
<<executable>>	Un archivo de programa ejecutable.
<<library>>	Una biblioteca estática o dinámica como una DLL o un archivo JAR.
<<script>>	Un script que se puede ejecutar por un intérprete.
<<source>>	Un archivo fuente que se puede compilar en un archivo ejecutable.

Puede esperar que varios perfiles UML se desplieguen con el tiempo para plataformas específicas de software como J2EE (Java 2 Platform, Enterprise Edition) y Microsoft .NET. Éstas proporcionarán un conjunto más rico de artefactos y otros estereotipos. La especificación UML 2.0 [UML2S] proporciona perfiles de ejemplo para J2EE y EJB, Microsoft COM, Microsoft .NET y CORBA (*Common Object Request Broker Architecture*, Arquitectura Común para Agentes de Petición de Objetos).

La tabla 24.2 muestra el perfil de ejemplo de Java.

Tabla 24.2.

Estereotipo	Se aplica a	Semántica
<<EJBEntityBean>>	Componente	Bean de entidad EJB.
<<EJBSessionBean>>	Componente	Bean de sesión EJB.
<<EJBMessageDrivenBean>>	Componente	Bean dirigido por mensaje EJB.
<<EJBHome>>	Interfaz	Interfaz EJB.
<<EJBRemote>>	Interfaz	Interfaz remota EJB.
<<EJBCreate>>	Operación	Operación crear EJB.
<<EJBBusiness>>	Operación	Una operación que soporta la lógica de negocio de la interfaz remota EJB.
<<EJBSecurityRoleRef>>	Asociación	Una asociación entre un EJB y un proveedor que proporciona una referencia a un rol de seguridad.
<<EJBRoleName>>	Actor	El nombre de un rol de seguridad.
<<EJBRoleNameRef>>	Actor	Una referencia a un rol de seguridad.
<<JavaSourceFile>>	Artefacto	Un archivo fuente Java.
<<JAR>>	Artefacto	Un archivo JAR.
<<EJBQL>>	Expresión	Una expresión en el lenguaje de consulta EJB.

Este perfil no es suficiente para modelar aplicaciones Java; falta un estereotipo para los archivos de clase Java y para los directorios. Ampliamos el perfil con los dos nuevos estereotipos listados en la tabla 24.3.

Tabla 24.3.

Estereotipo	Se aplica a	Semántica
`<<JavaClassFile>>`	Artefacto	Un archivo de clase Java (un archivo fuente Java compilado).
`<<directory>>`	Artefacto	Un directorio.

En la figura 24.8 hemos aplicado el perfil ampliado de Java a partir de la especificación UML para nuestro modelo. Como puede ver, el diagrama es mucho más significativo cuando aplica estereotipos descriptivos.

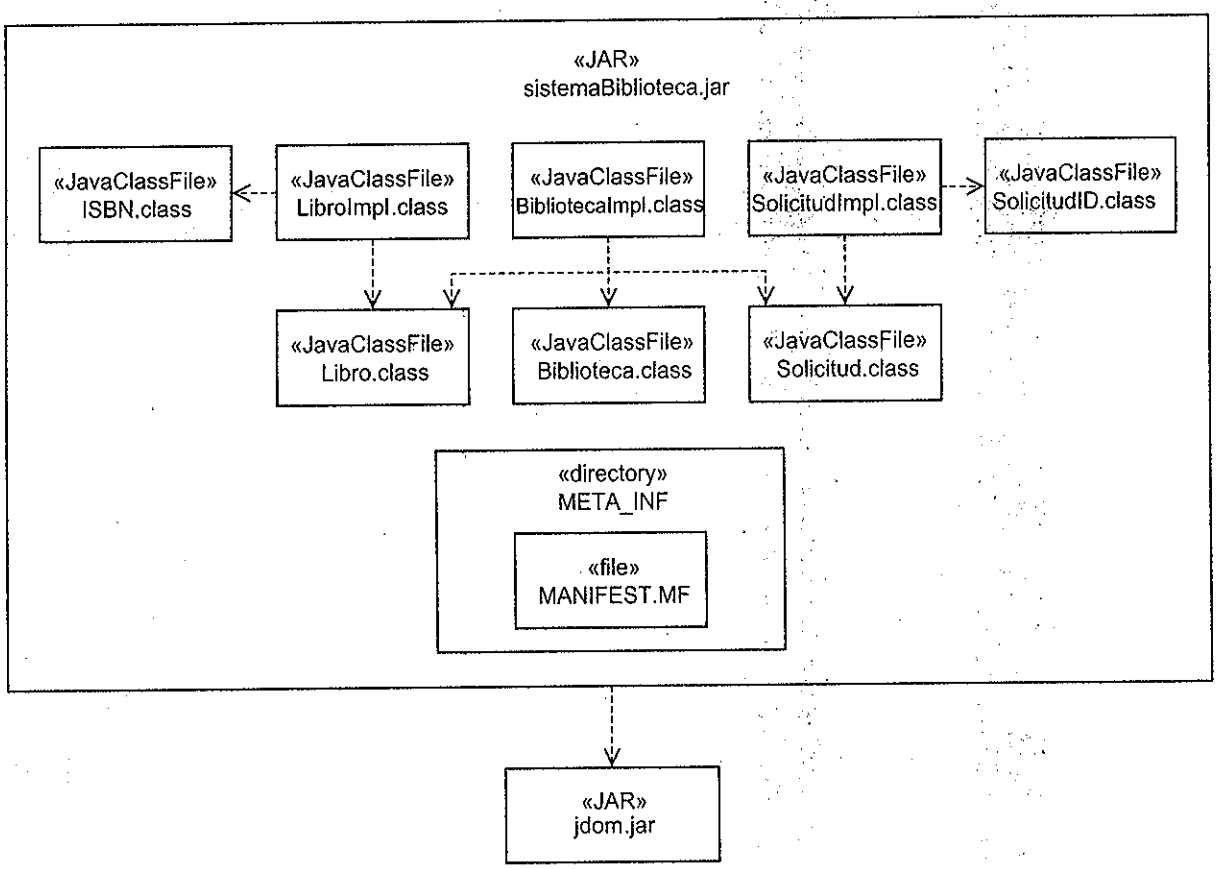


Figura 24.8.

24.6. Despliegue

Un sencillo diagrama de despliegue en forma de instancia se muestra en la figura 24.9.

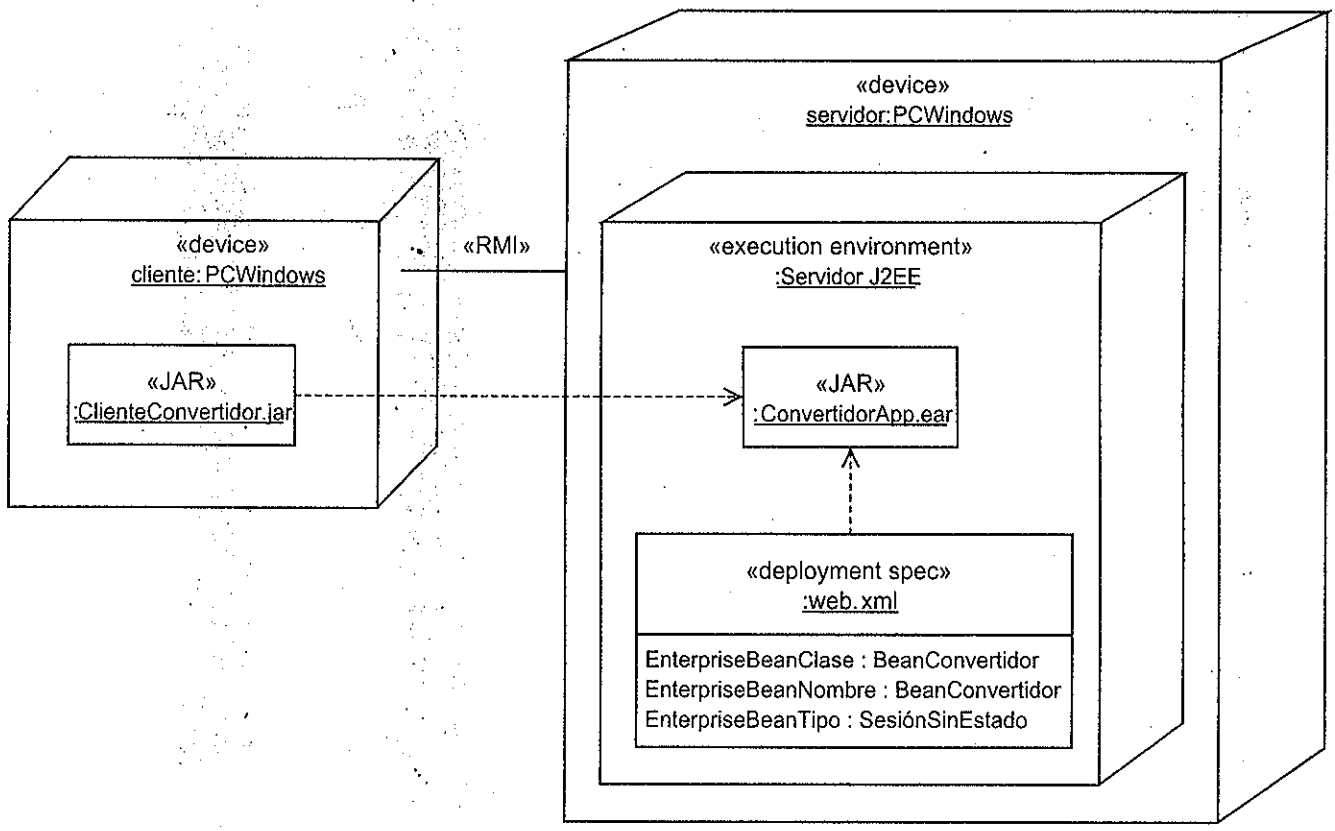


Figura 24.9.

Este ejemplo procede de un tutorial de Java (`www.java.sun.com`). Se trata de una aplicación de convertidor de moneda. La figura 24.9 muestra el archivo EAR (*Enterprise application Archive*) denominado `ConvertidorApp.ear` desplegado en un entorno de ejecución `Servidor J2EE` en un nodo denominado `servidor` del tipo `PCWindows`. El `Servidor J2EE` es un servidor de aplicación de Sun que se distribuye como parte de J2EE. Los archivos EAR son un tipo especial de archivo JAR que alberga aplicaciones J2EE. La aplicación del servidor de despliegue se utiliza por la aplicación cliente, `ClienteConvertertidor.jar`, que se ejecuta en un nodo denominado `cliente` que también es del tipo `PCWindows`.

Puede anexar una especificación de despliegue a un artefacto de despliegue como se muestra en la figura. La especificación de despliegue contiene detalles clave sobre el despliegue. En este caso, especificamos lo siguiente:

- `EnterpriseBeanClase`: Ésta es la clase que contiene la lógica del bean.
- `EnterpriseBeanNombre`: Éste es el nombre que los clientes pueden utilizar para acceder al bean.
- `EnterpriseBeanTipo`: Éste es el tipo del bean. En este caso, es un bean de sesión sin estado; un bean utilizado para transacciones sencillas que no tiene estado y no es persistente.

Existe mucho más en el despliegue EJB que lo que este sencillo ejemplo pudiera sugerir, y algunos de estos detalles dependen incluso del entorno de ejecución. Sin embargo, la finalidad de modelar en despliegue es capturar los detalles clave de despliegue y por lo tanto el descriptor de despliegue mostrado puede ser suficiente.

24.7. ¿Qué hemos aprendido?

Los diagramas de despliegue le permiten modelar la distribución de su sistema de software sobre hardware físico. Ha aprendido lo siguiente:

- La actividad UP `Implementación de arquitectura` trata sobre identificar componentes significativos arquitectónicamente y mapearlos con hardware físico.
- El diagrama de despliegue mapea la arquitectura de software a la arquitectura de hardware.
- En el workflow de diseño puede crear un diagrama de despliegue "inicial" al identificar nodos o instancias de nodos y relaciones; mejore esto como parte del workflow de implementación al añadir componentes o instancias de componente.
- El diagrama de despliegue de forma de descriptor se puede utilizar para modelar qué tipos de hardware, software y conexiones habrá en el sistema final desplegado.
  - Describe un conjunto completo de despliegues posibles.
  - Muestra:
    - Nodos: Qué tipos de hardware gestionan el sistema.
    - Relaciones: Los tipos de conexiones entre los nodos.
    - Componentes: Los tipos de componentes desplegados en nodos determinados.
- El diagrama de despliegue en forma de instancia muestra un despliegue determinado del sistema sobre piezas de hardware específicas e identificables.
  - Describe un despliegue específico del sistema, quizás en un sitio específico del usuario.
  - Muestra:
    - Instancias de nodo: Piezas específicas de hardware.
    - Instancias de relación: Relaciones específicas entre instancias de nodo.
    - Instancias de componente: Piezas específicas identificables de software desplegado en una instancia de nodo; por ejemplo, una copia determinada de Microsoft Office con un número de serie único.
- Nodo: Representa un tipo de recurso computacional.
  - `<<device>>`: Un tipo de dispositivo físico como un PC o un servidor Sun Fire.

    - <<execution environment>>: Un tipo de entorno de ejecución para software como un servidor Web Apache.
- Instancia de nodo: Representa un recurso computacional específico.
- Artefacto: Representa la especificación de un elemento del mundo real como un archivo ejecutable.
    - Los artefactos pueden mostrar uno o más componentes.
- Instancia de artefacto: Rrepresenta una instancia específica de un artefacto determinado, como una copia específica de un archivo ejecutable determinado desplegado en una máquina determinada.