dividuales en bloques arquitectónicos más grandes y representar las interfaces (mecanismos de conexión) que hay entre los componentes. Una vez establecidas las técnicas descriptivas basadas en lenguaje para el diseño de la arquitectura, es más probable que, a medida que el diseño evoluciona, se obtengan métodos de evaluación eficaces para las arquitecturas.

9.6 MAPEO DE LA ARQUITECTURA CON EL USO DEL FLUJO DE DATOS

Los estilos arquitectónicos analizados en la sección 9.3.1 representan arquitecturas radicalmente distintas. Por ello, no sorprende que no exista un mapeo exhaustivo que efectúe la transición del modelo de requerimientos a varios estilos de arquitectura. En realidad, no hay un mapeo práctico para ciertos estilos y el diseñador debe enfocar la traducción de los requerimientos a su diseño con el empleo de las técnicas descritas en la sección 9.4.

Para ilustrar un enfoque al mapeo arquitectónico, considere la arquitectura denominada de llamada y regreso, estructura muy común para muchos tipos de sistemas. La arquitectura de llamada y regreso reside dentro de otras más sofisticadas que ya se analizaron en este capítulo. Por ejemplo, la arquitectura de uno o más componentes de una arquitectura cliente-servidor podría denominarse de llamada y regreso.

Una técnica de mapeo llamada *diseño estructurado* [You79] se caracteriza con frecuencia como método de diseño orientado al flujo porque provee una transición conveniente de un diagrama de flujo de datos (véase el capítulo 7) a la arquitectura del software.[7] La transición del flujo de la información (representada con el diagrama de flujo de datos o DFD) a estructura de programa se consigue como parte de un proceso de seis pasos: 1) se establece el tipo de flujo de información, 2) se indican las fronteras del flujo, 3) se mapea el DFD en la estructura del programa, 4) se define la jerarquía del control, 5) se refina la estructura resultante con el empleo de criterios de medición para el diseño y heurísticos, y 6) se mejora y elabora la descripción arquitectónica.

Como ejemplo breve del mapeo de flujo de datos, se presenta uno de "transformación" paso a paso para una pequeña parte de la función de seguridad *CasaSegura*.[8] Con objeto de realizar el mapeo, debe determinarse el tipo de flujo de la información. Un tipo de flujo de información se llama *flujo de transformación* si presenta una cualidad lineal. Los datos fluyen al sistema con una *trayectoria de flujo de entrada* en la que se transforman de una representación del mundo exterior a una forma internalizada. Una vez internalizados, se procesan en un *centro de transformación*. Por último, salen del sistema por una *trayectoria de flujo de salida* que transforma los datos a una forma del mundo exterior.[9]

9.6.1 Mapeo de transformación

El mapeo de transformación es un conjunto de pasos de diseño que permite mapear un DFD con características de flujo de transformación en un estilo arquitectónico específico. Para ilustrar este enfoque, de nuevo consideraremos la función de seguridad de *CasaSegura*.[10] Un elemento del modelo de análisis es un conjunto de diagramas de flujo de datos que describen el flujo de información dentro de la función de seguridad. Para mapear estos diagramas de flujo de datos en una arquitectura de software deben darse los siguientes pasos de diseño:

---

7 Debe observarse que durante el método de mapeo también se utilizan otros elementos del modelo de requerimientos.

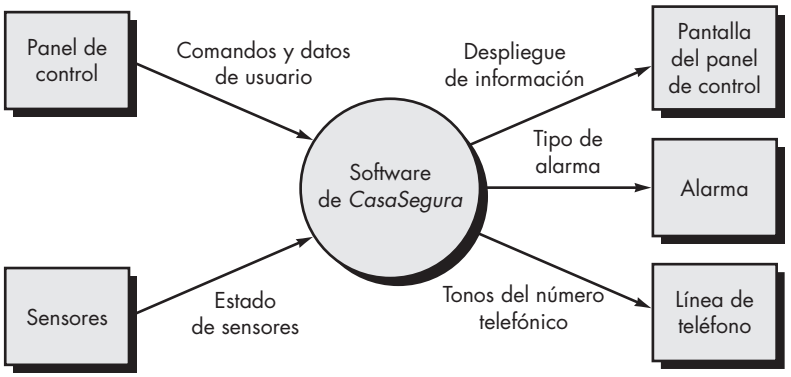
8 En el sitio web del libro se presenta un análisis más detallado del diseño estructurado.

9 En este ejemplo no se considera otro tipo importante de flujo de información, pero se aborda en el ejemplo de diseño estructurado que se presenta en el sitio web del libro.

10 Sólo se considera la parte de la función de seguridad de *CasaSegura* que utiliza el panel de control. Aquí no se incluyen otras características analizadas en el libro.

FIGURA 9.10

Diagrama de flujo de datos para la función de seguridad de *CasaSegura*



Paso 1. Revisión del modelo del sistema fundamental. El modelo del sistema fundamental o diagrama de contexto ilustra la función de seguridad como una transformación única, y representa a los productores y consumidores externos de los datos que fluyen hacia dentro y fuera de la función. La figura 9.10 ilustra un modelo de contexto de nivel 0, y la figura 9.11 muestra el flujo de datos refinado para la función de seguridad.

CONSEJO

Si en este momento se mejora más el diagrama de flujo de datos, trate de obtener burbujas que presenten mucha cohesión.

Paso 2. Revisar y mejorar los diagramas de flujos de datos para el software. La información obtenida del modelo de requerimientos se refina para producir más detalles. Por ejemplo, se estudia el DFD de nivel 2 para *vigilar sensores* (véase la figura 9.12) y se obtiene el diagrama de flujo de datos de nivel 3 que se presenta en la figura 9.13. En el nivel 3, cada transformación en el diagrama de flujo de datos presenta una cohesión relativamente grande (consulte el capítulo 8). Es decir, el proceso implícito por una transformación realiza una sola y distintiva función que se implementa como componente en el software de *CasaSegura*. Entonces, el DFD de la figura 9.13 contiene detalles suficientes para hacer un "primer corte" en el diseño de la arquitectura del subsistema *vigilar sensores*, y se continúa con más refinamiento.

FIGURA 9.11

Diagrama de flujo de datos de nivel 1 para la función de seguridad de *CasaSegura*

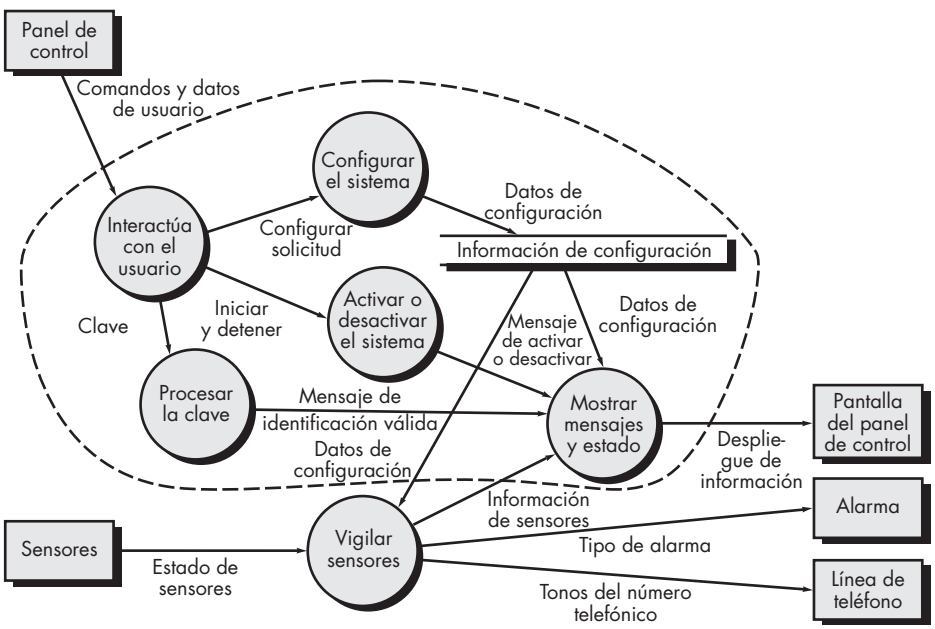
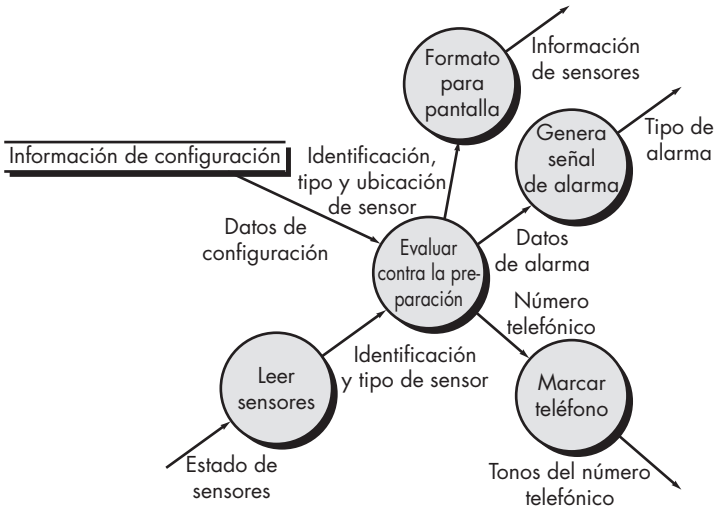


FIGURA 9.12

Diagrama de flujo de datos de nivel 2 que mejora la transformación *vigilar sensores*



PUNTO CLAVE

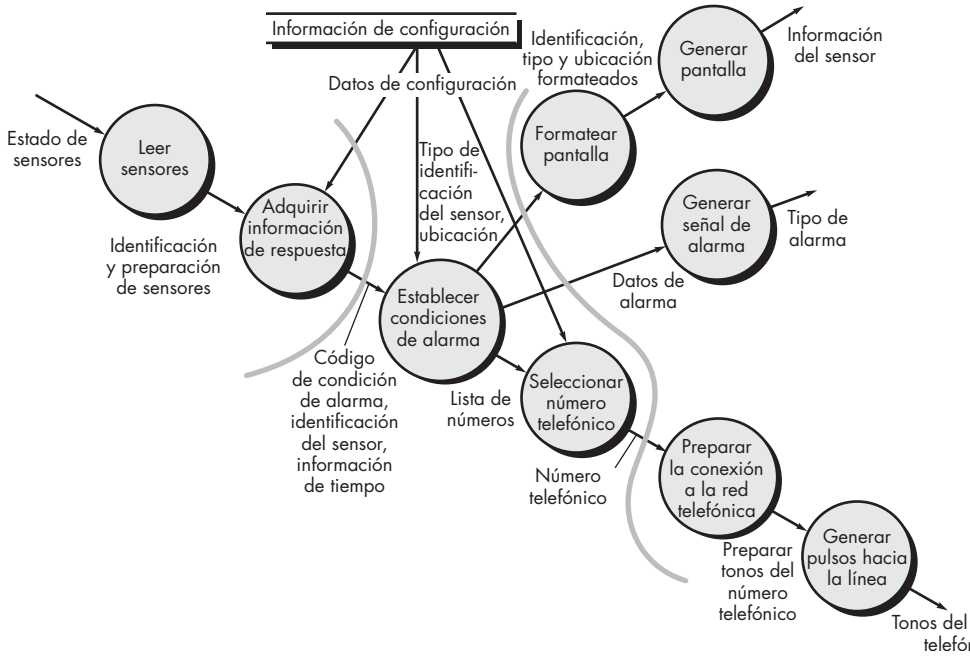
Será frecuente encontrar varios tipos de flujo de datos dentro del mismo modelo orientado al flujo. Los flujos se dividen y la estructura del programa se obtiene con el uso del mapeo apropiado.

Paso 3. Determinar si el DFD tiene características de flujo de transformación o de transacción.[11] Al evaluar el DFD (véase la figura 9.13) se observa que los datos entran al software por una trayectoria de ingreso y lo abandonan por tres trayectorias de salida. Por tanto, se adoptará una característica general de transformación para el flujo de la información.

Paso 4. Aísle el centro de transformación, especificando las fronteras de entrada y salida del flujo. Los datos de entrada fluyen por una trayectoria en la que la información pasa de su forma externa a su forma interna; el flujo de salida convierte los datos internalizados a su forma externa. Las fronteras de los flujos de entrada y salida quedan abiertas a la interpretación.

FIGURA 9.13

Diagrama de flujo de datos de nivel 3 para *vigilar sensores* con fronteras del flujo



11 En el flujo de transacción, un solo concepto de datos, llamado *transacción*, ocasiona que el flujo de datos se ramifique a través de cierto número de trayectorias definidas por la naturaleza de la transacción.

En un esfuerzo por explorar estructuras alternativas para el programa, varíe las fronteras del flujo. Esto toma poco tiempo y da una perspectiva importante.

Es decir, diferentes diseñadores tal vez seleccionen como ubicación de la frontera diferentes puntos en el flujo. De hecho, es posible obtener soluciones alternativas al diseño si se varía la colocación de las fronteras del flujo. Aunque debe tenerse cuidado al seleccionar las fronteras, la variación de una burbuja a lo largo de la trayectoria del flujo tendrá por lo general poco efecto en la estructura final del programa.

Para el ejemplo, en la figura 9.13 se ilustran las fronteras del flujo como curvas sombreadas que corren de arriba abajo a través del flujo. Las transformaciones (burbujas) que constituyen el centro de transformación quedan dentro de esas dos fronteras sombreadas. Es posible dar argumentos para reajustar una frontera (por ejemplo, podría proponerse una frontera para el flujo de entrada que separara *leer sensores* de *adquirir información de respuesta*). En este diseño, el énfasis en esta etapa de diseño debe estar en la selección de fronteras razonables y no en la iteración extensa en la colocación de las divisiones.

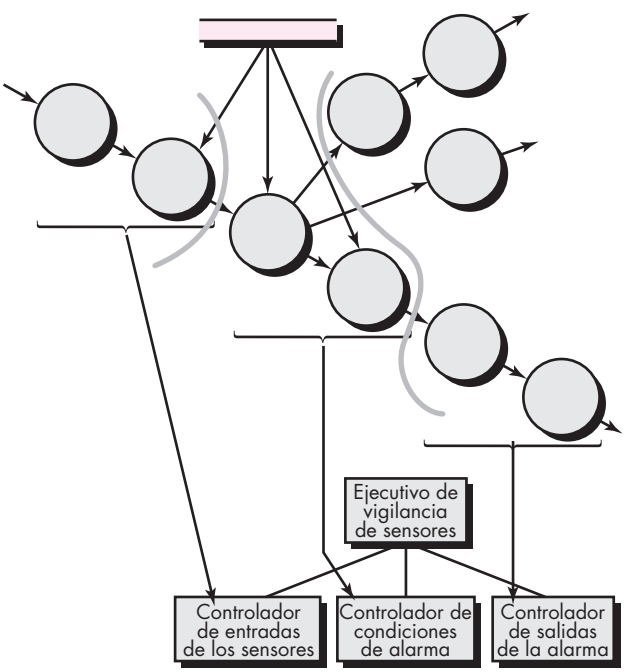
Paso 5. Realizar el "rediseño de primer nivel". La arquitectura del programa obtenida con este mapeo da como resultado una distribución del control de arriba abajo. El *rediseño* lleva a una estructura de programa en la que los componentes de alto nivel ejecutan la toma de decisiones y los de bajo nivel realizan la mayor parte del trabajo de entrada, computación y salida. Los componentes de nivel medio llevan a cabo cierto control y moderan las cantidades de trabajo.

Cuando se encuentra una transformación, se mapea un diagrama de flujo de datos para una estructura específica (una de llamar y regresar) que provea control para el procesamiento de información de entrada, transformación y salida. En la figura 9.14 se ilustra este rediseño de primer nivel para el subsistema *vigilar sensores*. Un controlador principal (llamado *ejecutivo de vigilancia de sensores*) reside en la parte superior de la estructura del programa y coordina las siguientes funciones de control subordinadas:

- Un controlador del procesamiento de la información de entrada, llamado *controlador de entradas de los sensores*, coordina la recepción de todos los datos que llegan.
- Un controlador del flujo de transformación, llamado *controlador de condiciones de la alarma*, supervisa todas las operaciones sobre los datos en forma internalizada (por ejemplo, un módulo que invoque varios procedimientos de transformación de datos.

FIGURA 9.14

Rediseño de primer nivel para *vigilar sensores*



En esta etapa no se debe ser dogmático. Tal vez sea necesario establecer dos o más controladores para el procesamiento de las entradas o la computación, con base en la complejidad del sistema que se va a elaborar. Si el sentido común sugiere este enfoque, ¡adelante!

- Un controlador de procesamiento de información de salida, llamado *controlador de salidas de la alarma*, coordina la producción de información de salida.

Aunque la figura 9.14 sugiere una estructura de tres dientes, los flujos complejos que hay en sistemas grandes tal vez requieran dos o más módulos de control para cada una de las funciones de control generales descritas con anterioridad. El número de módulos en el primer nivel debe limitarse al mínimo necesario para ejecutar las funciones de control y mantener buenas características de independencia de funciones.

Paso 6. Realizar "rediseño de segundo nivel". El rediseño de segundo nivel se logra con el mapeo de transformaciones individuales (burbujas) de un diagrama de flujo de datos en módulos apropiados dentro de la arquitectura. Se comienza en la frontera del centro de transformación y se avanza hacia afuera a lo largo de las trayectorias de entrada y salida; las transformaciones se mapean en niveles subordinados de la estructura del software. En la figura 9.15 se presenta el enfoque general del rediseño de segundo nivel.

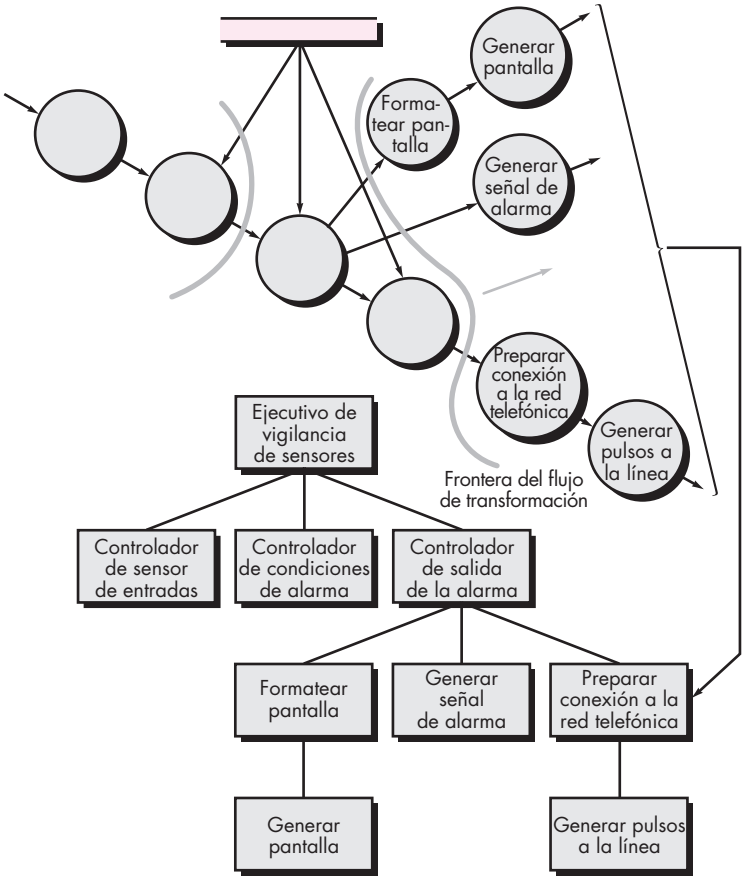
Hay que eliminar los módulos de control redundantes. Es decir, si un módulo de control no hace nada más que controlar a otro módulo, su función controladora debe llevarse a un módulo de nivel más alto.

Aunque la figura 9.15 ilustra un mapeo uno a uno entre las transformaciones del diagrama de flujo y los módulos de software, es frecuente que haya distintos mapeos. Es posible combinar y representar dos o incluso tres burbujas como un solo componente, o una sola burbuja puede expandirse a dos o más componentes. Son consideraciones prácticas y mediciones de calidad del diseño las que dictan el resultado del rediseño de segundo nivel. La revisión y refinamiento tal vez produzcan cambios en esta estructura, pero sirven como diseño de "primera iteración".

El rediseño de segundo nivel para el flujo de entrada se presenta de la misma manera. El rediseño se logra de nuevo avanzando hacia afuera a partir de la frontera del centro de trans-

FIGURA 9.15

Rediseño de segundo nivel para *vigilar sensores*



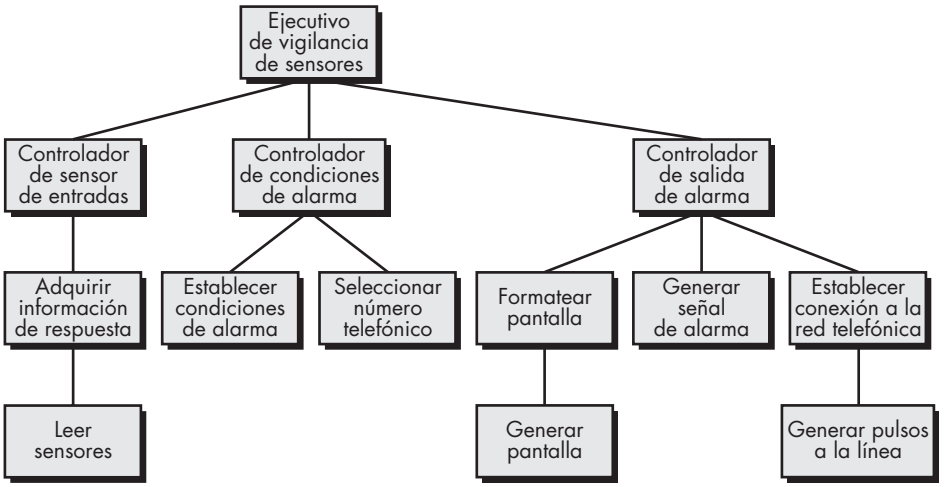


FIGURA 9.16
Estructura de primera iteración para *vigilar sensores*

CONSEJO

Conserve módulos "trabajadores" en un nivel bajo de la estructura del programa. Esto llevará a una arquitectura fácil de mantener.

formación en el lado del flujo de entrada. El centro de transformación del software del subsistema *vigilar sensores* se mapea de modo un poco distinto. Cada conversión de datos o transformación de cálculo de la parte de transformación del diagrama de flujo de datos se mapea en un módulo subordinado al controlador de la transformación. En la figura 9.16 se presenta la arquitectura completa de primera iteración.

Los componentes así mapeados que se aprecian en la figura 9.16 representan un diseño inicial de la arquitectura del software. Aunque los componentes reciben su nombre de manera que se implique la función, para cada uno debe escribirse una narración breve (adaptada de la especificación del proceso desarrollada para la transformación de datos y que se generó durante el modelado de los requerimientos). La narración describe la interfaz del componente, las estructuras internas de los datos, un relato de las funciones y el análisis breve de las restricciones y otros rasgos especiales (como los archivos de entrada y salida, características que dependen del hardware, requerimientos especiales de tiempo, etcétera).

Cita:

"Hacerlo tan sencillo como sea posible. Pero no más."

Albert Einstein

Paso 7. Refinar la arquitectura de primera iteración con el empleo de heurísticos de diseño para mejorar la calidad del software. Siempre es posible refinar la arquitectura de primera iteración, aplicando conceptos de independencia de funciones (véase el capítulo 8). Los componentes hacen explosión o implosión para producir un rediseño sensible, la separación de problemas, buena cohesión, acoplamiento mínimo y, lo más importante, una estructura que puede implementarse sin dificultad, probar sin confusión y mantener sin grandes problemas.

Los refinamientos son impuestos por el análisis y los métodos de evaluación descritos en la sección 9.5, así como por consideraciones prácticas y el sentido común. Por ejemplo, hay ocasiones en las que el controlador para el flujo de datos de entrada es totalmente innecesario, cuando se requiere algún procesamiento de las entradas en un componente subordinado al controlador de la transformación, cuando no es posible evitar mucho acoplamiento debido a datos globales o cuando no se logran características estructurales óptimas. El arbitraje final lo constituyen los requerimientos del software acoplados con el criterio humano.

El objetivo de los siete pasos anteriores es desarrollar una representación arquitectónica del software. Es decir, una vez definida la estructura, se evalúa y mejora considerándola como un todo. Las modificaciones que se hacen en este momento exigen poco trabajo adicional, pero tienen un efecto profundo en la calidad del software.

Debe hacerse una pausa y considerar la diferencia entre el enfoque de diseño descrito y el proceso de "escribir programas". Si el código es la única representación del software, usted y

CasaSegura

Refinación de la arquitectura de primer corte

La escena: El cubículo de Jamie, cuando comienza la modelación del diseño.

Participantes: Jamie y Ed, miembros del equipo de ingeniería de software de *CasaSegura*.

La conversación:

[Ed acaba de terminar el diseño de primer corte del subsistema de vigilancia de sensores. Se detiene para solicitar la opinión de Jamie.]

Ed: Pues bien, aquí está la arquitectura que obtuve.

[Ed muestra a Jamie la figura 9.16, y ella la estudia unos momentos.]

Jamie: Está muy bien, pero creo que podemos hacer algo para que sea más sencilla… y mejor.

Ed: ¿Como qué?

Jamie: Bueno, ¿por qué usaste el componente *controlador de sensores de entrada*?

Ed: Porque se necesita un controlador para el mapeo.

Jamie: No en realidad. El controlador no hace gran cosa, ya que estamos manejando una sola trayectoria de flujo para los datos de entrada. Puede eliminarse el controlador sin que pase nada.

Ed: Puedo vivir con eso. Lo cambiaré y…

Jamie (sonríe): Espera… También podemos hacer la implosión de los componentes *establecer condiciones de alarma* y *seleccionar número telefónico*. El controlador de la transformación que presentas en realidad no es necesario y la poca disminución en la cohesión es tolerable.

Ed: Simplificación, ¿eh?

Jamie: Sí. Y al hacer refinamientos sería buena idea hacer la implosión de los componentes *formatear pantalla* y *generar pantalla*. El formateo de la pantalla para el panel de control es algo sencillo. Puede definirse un nuevo módulo llamado *producir pantalla*.

Ed (dibuja): Entonces, ¿esto es lo que piensas que debemos hacer?

[Muestra a Jamie la figura 9.17.]

Jamie: Es un buen comienzo.

sus colegas tendrán grandes dificultades para evaluarlo o mejorarlo en un nivel global u holístico y, en verdad, tendrán dificultades porque "los árboles no los dejarán ver el bosque".

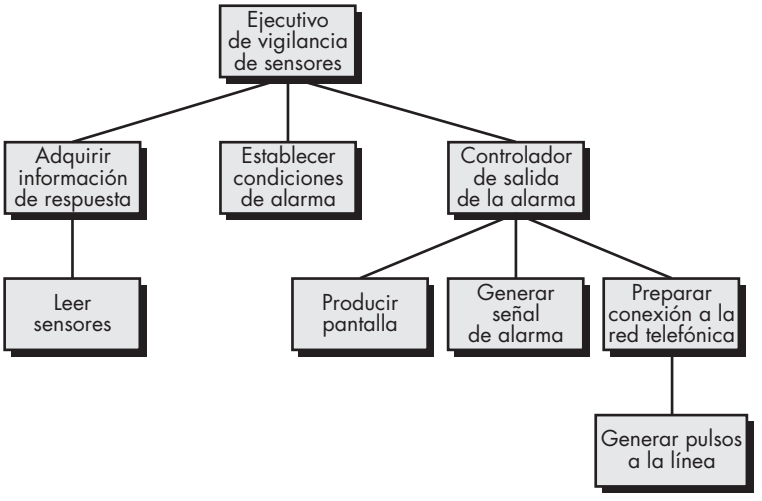
9.6.2 Refinamiento del diseño arquitectónico

? ¿Qué pasa después de que se generó la arquitectura?

Cualquier análisis del refinamiento del diseño debería ir precedido de este comentario: "Recuerde que un 'diseño óptimo' que no funcione tiene un mérito cuestionable." Debe ocuparse de desarrollar una representación del software que satisfaga todos los requerimientos funcionales y de desempeño, y darle mérito según sus mediciones y heurísticos de diseño.

Debe estimularse el refinamiento de la arquitectura del software durante las primeras etapas del diseño. Como ya se dijo en este capítulo, hay estilos alternativos para la arquitectura que es posible obtener, refinar y evaluar en busca del "mejor" enfoque. Éste, dirigido a la optimización,

Figura 9.17
Estructura refinada del programa para *vigilar sensores*



es uno de los verdaderos beneficios que se logran con el desarrollo de una representación de la arquitectura del software.

Es importante observar que la sencillez estructural con frecuencia refleja tanto elegancia como eficiencia. El refinamiento del diseño debe perseguir el menor número de componentes que sea consistente con la modularidad efectiva y la estructura de datos menos compleja que cumpla de modo adecuado con los requerimientos de información.

9.7 Resumen

La arquitectura del software proporciona una visión holística del sistema que se va a construir. Ilustra la estructura y organización de los componentes del software, sus propiedades y conexiones. Los componentes del software incluyen módulos de programa y las distintas representaciones de datos que manipula éste. Por tanto, el diseño de los datos es parte integral de la generación de la arquitectura del software. Ésta subraya las primeras decisiones respecto del diseño y provee un mecanismo para considerar los beneficios de las estructuras alternativas para el sistema.

Dentro de un género arquitectónico dado, hay varios estilos y patrones diferentes disponibles para el ingeniero de software. Cada estilo describe una categoría de sistemas que agrupa un conjunto de componentes que realizan una función requerida por el sistema; un grupo de conectores que permiten comunicación, coordinación y cooperación entre los componentes; restricciones que definen cómo pueden integrarse éstos para formar el sistema y modelos semánticos que permiten que un diseñador entienda las propiedades generales del sistema.

En un sentido general, el diseño arquitectónico se obtiene con el empleo de cuatro pasos. En primer lugar, el sistema debe representarse en contexto. Es decir, el diseñador debe definir las entidades externas con las que interactúa el software y la naturaleza de la interacción. Una vez especificado el contexto, el diseñador debe identificar un conjunto de abstracciones de alto nivel, llamadas *arquetipos*, que representan elementos fundamentales del comportamiento o función del sistema. Ya que se definieron las abstracciones, el diseño comienza a avanzar cerca del dominio de la implementación. Se identifican los componentes y se representan dentro del contexto de una arquitectura que les da apoyo. Por último, se desarrollan instancias específicas de la arquitectura para "probar" el diseño en el contexto del mundo real.

Como ejemplo sencillo del diseño arquitectónico, el método del mapeo presentado en este capítulo usa las características del flujo de datos para obtener un estilo arquitectónico de uso muy común. El diagrama de flujo de datos se mapea en la estructura del programa con el uso del enfoque del mapeo de la transformación. Éste se aplica a un flujo de información que presente fronteras distintas entre los datos de entrada y los de salida. El diagrama de flujo de datos se mapea en una estructura que asigna el control a la entrada, al procesamiento y a la salida junto con tres jerarquías de módulos diseñados por separado. Una vez que se tiene la arquitectura, se elabora y analiza mediante criterios de calidad.

Problemas y puntos por evaluar

9.1. Con el uso de la arquitectura de una casa o edificio como metáfora, establezca comparaciones con la arquitectura del software. ¿En qué se parecen las disciplinas de la arquitectura clásica y la del software? ¿En qué difieren?

9.2. Diga dos o tres ejemplos de aplicaciones para cada uno de los estilos arquitectónicos mencionados en la sección 9.3.1.

9.3. Algunos de los estilos arquitectónicos citados en la sección 9.3.1 tienen naturaleza jerárquica, mientras que otros no. Elabore una lista de cada tipo. ¿Cómo se implementarían los estilos arquitectónicos que no son jerárquicos?