

# ELABORACIÓN DE PROTOTIPOS, RAD Y PROGRAMACIÓN EXTREMA

# 6

## OBJETIVOS DE APRENDIZAJE

Una vez que haya dominado el material de este capítulo, podrá:

1. Entender los cuatro modelos principales de elaboración de prototipos.
2. Usar la elaboración de prototipos para la recopilación de los requerimientos de información.
3. Comprender el concepto de RAD para usarlo en la recopilación de requerimientos de información y el diseño de interfaces.
4. Entender la programación extrema y las prácticas esenciales que lo diferencian de otras metodologías de desarrollo.
5. Apreciar la importancia de los valores que son críticos para la programación extrema y la modelación ágil.

La elaboración de prototipos de sistemas de información es una técnica valiosa para recopilar rápidamente datos específicos sobre los requerimientos de información de los usuarios. En términos generales, la elaboración de prototipos eficaz debe realizarse en las primeras etapas del ciclo de vida del desarrollo de sistemas, durante la fase de determinación de requerimientos. Sin embargo, la elaboración de prototipos es una técnica compleja que requiere conocimiento de todo el ciclo de vida del desarrollo de sistemas para completarse con éxito.

La elaboración de prototipos se incluye en este punto del texto para subrayar su importancia como una técnica de recopilación de información. Cuando la elaboración de prototipos se usa de esta forma, lo que el analista de sistemas busca son las primeras reacciones hacia el prototipo por parte de los usuarios y los directivos, las sugerencias del usuario sobre cambiar o limpiar el sistema del cual se elaboró el prototipo, sus posibles innovaciones y los planes de revisión que detallen las partes del sistema que se necesitan hacer primero o de cuáles divisiones de una organización se hará el próximo prototipo.

Un tipo especial de elaboración de prototipos que usa un enfoque orientado a objetos se llama *desarrollo rápido de aplicaciones*, o RAD. La elaboración de prototipos y RAD también se pueden usar como un método alternativo al SDLC.

## ELABORACIÓN DE PROTOTIPOS

Como analista de sistemas que presenta un prototipo del sistema de información, usted está bastante interesado en las reacciones de los usuarios y los directivos de la organización hacia el prototipo. Usted desea saber detalladamente cómo reaccionarán al trabajar con el prototipo y qué tan bien satisfarán sus necesidades las características del sistema a partir de las cuales se elab-

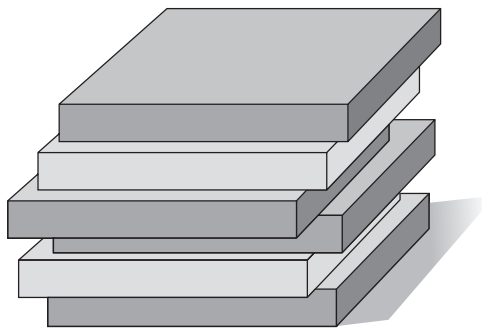
boró el prototipo. Las reacciones se recopilan a través de la observación, las entrevistas y las hojas de retroalimentación (posiblemente los cuestionarios) diseñados para obtener la opinión de cada persona sobre el prototipo después de que interactúan con él.

La información recopilada en la fase de elaboración de prototipos permite al analista establecer las prioridades y cambiar el rumbo de los planes a bajo costo, con un mínimo de molestias. Debido a esta característica, la elaboración de prototipos y la planeación van de la mano.

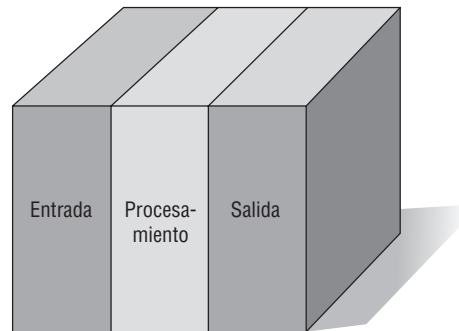
## CLASES DE PROTOTIPOS

La palabra *prototipo* se usa de muchas formas diferentes. En lugar de intentar sintetizar todos estos usos en una sola definición o de tratar de convenir en un enfoque correcto al tema un tanto polémico de la elaboración de prototipos, ilustramos la manera en que cada una de varias concepciones de la elaboración de prototipos se puede aplicar convenientemente en una situación particular, como se muestra en la figura 6.1.

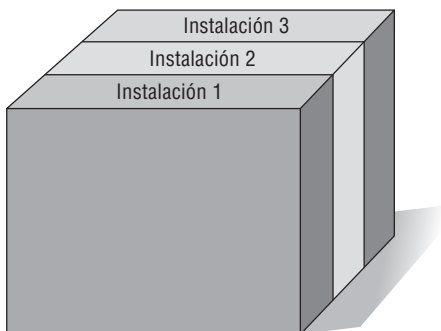
**Prototipo corregido** La primera clase de elaboración de prototipos tiene que ver con la construcción de un sistema que funciona pero se corrige simultáneamente. En la ingeniería a este enfoque se le llama elaboración de una tabla experimental: la creación, en una tableta de pruebas, de un modelo funcional de un circuito integrado (que en la vida real sería microscópico).



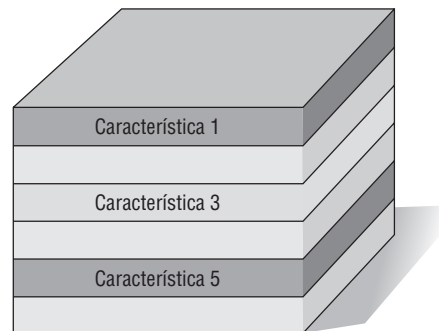
Prototipo corregido



Prototipo no funcional



Primer prototipo de una serie



Prototipo de características seleccionadas

### FIGURA 6.1

Cuatro tipos de prototipos (en el sentido de las manecillas del reloj, a partir del lado superior izquierdo).

Un ejemplo en sistemas de información es un modelo funcional que tiene todas las características necesarias pero es ineficiente. En este ejemplo de elaboración de prototipos, los usuarios pueden interactuar con el sistema, acostumbrándose a la interfaz y los tipos de salidas disponibles. Sin embargo, la recuperación y almacenamiento de información podrían ser ineficientes, debido a que los programas se escribieron rápidamente con el objetivo de ser funcionales en lugar de eficaces.

**Prototipo no funcional** El segundo tipo de prototipo es un modelo no funcional a escala configurado para probar ciertos aspectos del diseño. Un ejemplo de este enfoque es un modelo a escala completa de un automóvil que se usa para pruebas en un túnel de viento. El tamaño y forma del automóvil son precisos, pero el automóvil no es funcional. En este caso sólo se incluyen las características del automóvil que son fundamentales para la prueba en el túnel de viento.

Un modelo no funcional a escala de un sistema de información podría producirse cuando la codificación requerida por las aplicaciones es demasiado extensa para incluirse en el prototipo pero cuando se puede conseguir una idea útil del sistema a través de la elaboración de un prototipo de la entrada y la salida. En este caso, el procesamiento, debido al excesivo costo y el tiempo requerido, no podría incluirse en el prototipo. Sin embargo, aún se podrían tomar algunas decisiones sobre la utilidad del sistema con base en la entrada y la salida incluidas en el prototipo.

**Primer prototipo de una serie** Un tercer tipo de prototipos involucra la creación de un primer modelo a escala completa de un sistema, con frecuencia llamado piloto. Un ejemplo es la elaboración de un prototipo del primer avión de una serie. El prototipo es completamente funcional y es una materialización de lo que el diseñador espera será una serie de aviones con características idénticas.

Este tipo de elaboración de prototipos es útil cuando se planean muchas instalaciones del mismo sistema de información. El modelo funcional a escala completa permite a los usuarios experimentar la interacción real con el nuevo sistema, pero minimiza el costo de superar cualquier problema que se presente. La creación de un modelo funcional es uno de los tipos de elaboración de prototipos que se hace con RAD, tratado más adelante en este capítulo.

Por ejemplo, cuando una cadena de tiendas de abarrotes minoristas considera el uso del EDI (intercambio electrónico de datos) para comprobar los envíos de los proveedores a varias tiendas, se podría instalar un modelo a escala completa en una tienda para resolver cualquier problema antes de que el sistema se implemente en todas las demás tiendas. Otro ejemplo es el de las instalaciones bancarias para la transferencia electrónica de fondos. Primero, se instala un prototipo a escala completa en una o dos sucursales, y si tiene éxito, se instalan los duplicados en todas las sucursales con base en los patrones de uso de los clientes y en otros factores importantes.

**Prototipo de características seleccionadas** Una cuarta concepción de la elaboración de prototipos involucra la creación de un modelo funcional que incluya algunas, pero no todas, de las características que tendrá el sistema final. Una analogía sería que un nuevo centro comercial minorista abriera antes de que se terminara la construcción de todas las tiendas.

Cuando se elaboran prototipos de los sistemas de información de esta manera, se incluyen algunas de las características principales, aunque no todas. Por ejemplo, en la pantalla podría aparecer un menú del sistema que muestre seis características: agregar un registro, actualizar un registro, eliminar un registro, buscar una palabra clave en un registro, listar un registro o examinar un registro. Sin embargo, en el prototipo del sistema tal vez sólo estén disponibles tres de las seis características, de manera que el usuario podría agregar un registro (característica 1), eliminar un registro (característica 3) y listar un registro (característica 5).

Cuando se recurre a este tipo de elaboración de prototipos, el sistema se completa por módulos de forma que si las características que se incluyen en los prototipos se evalúan exitosamente, se puedan incorporar en el sistema final más grande sin necesidad de realizar demasiado esfuerzo en la interacción. Los prototipos hechos de esta forma son parte del sistema real. No son sólo un modelo como en el caso de los prototipos no funcionales que se describieron antes.



Fuente: © Tedd Goff 2003 de cartoonbank.com. Todos los derechos reservados.

"Prefiero los íconos"

### **ELABORACIÓN DE PROTOTIPOS COMO UNA ALTERNATIVA AL CICLO DE VIDA DEL DESARROLLO DE SISTEMAS**

Algunos analistas argumentan que la elaboración de prototipos se debe considerar como una alternativa para el ciclo de vida del desarrollo de sistemas (SDLC). Recuerde que el SDLC, tratado en el capítulo 1, es un enfoque lógico y sistemático que se sigue en el desarrollo de sistemas de información.

Las quejas relativas al proceso del SDLC se centran en dos preocupaciones interrelacionadas. La primera preocupación es todo el tiempo que se requiere para pasar por el ciclo de vida del desarrollo. Conforme aumenta la inversión de tiempo del analista, el costo del sistema entregado se incrementa proporcionalmente.

La segunda preocupación sobre el uso del SDLC es que los requerimientos del usuario cambian a través del tiempo. Los requerimientos del usuario evolucionan durante el considerable intervalo existente entre el análisis de los requerimientos del usuario y la fecha en que se entrega el sistema final. Por lo tanto, debido al extenso ciclo del desarrollo, el sistema resultante podría ser criticado por abordar deficientemente los requerimientos de información del usuario actual.

Un corolario al problema de mantenerse al tanto de los requerimientos de información del usuario es la teoría de que los usuarios realmente no saben lo que hacen o no lo desean sino hasta que ven algo tangible. En el SDLC tradicional, una vez que se entrega un sistema, con frecuencia es demasiado tarde para modificarlo.

Para resolver estos problemas, algunos analistas proponen la elaboración de prototipos como una alternativa al ciclo de vida del desarrollo de sistemas. Cuando la elaboración de prototipos se usa de esta forma, el analista reduce efectivamente el tiempo entre la determinación de los requerimientos de información y la entrega de un sistema funcional. Además, el uso de la elaboración de prototipos en lugar del SDLC tradicional podría resolver algunos problemas como el de identificar con precisión los requerimientos de información del usuario.

Entre las desventajas de sustituir el SDLC por la elaboración de prototipos está la de la configuración prematura de un sistema antes de que el problema u oportunidad en cuestión se entienda completamente. También, el uso de la elaboración de prototipos como una

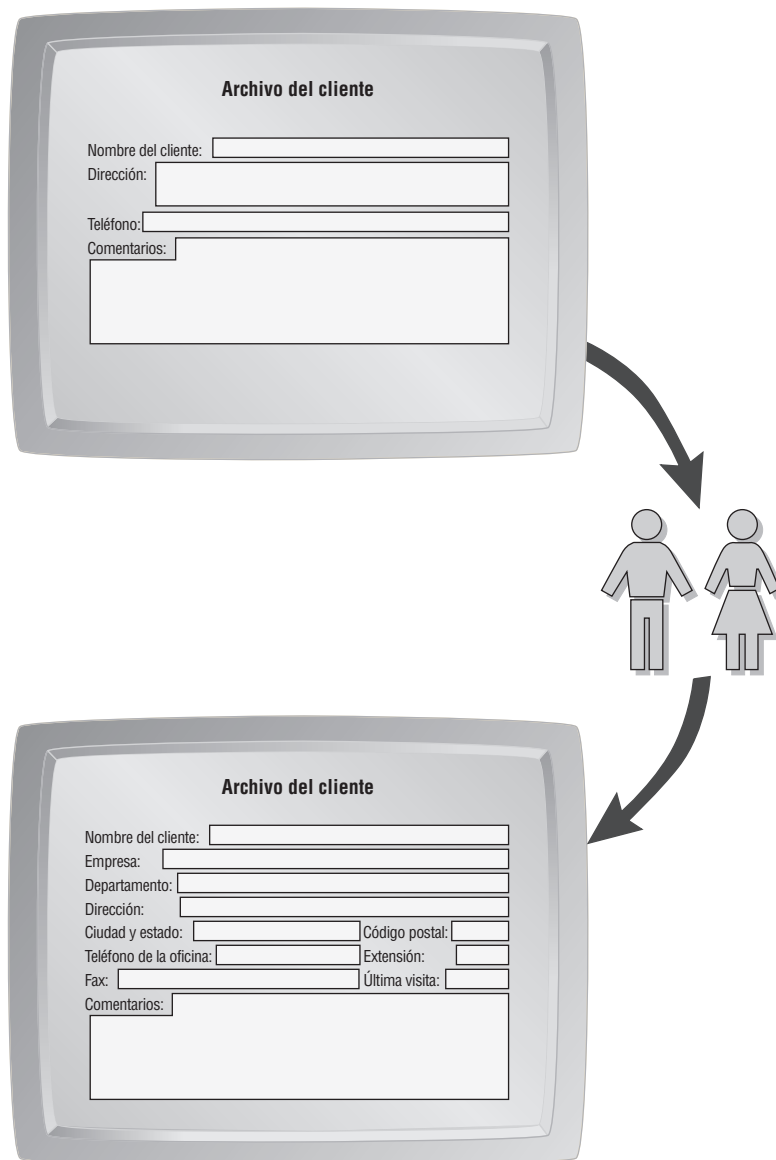
alternativa podría producir un sistema aceptado por grupos específicos de usuarios pero inadecuado para las necesidades globales del sistema.

El enfoque que apoyamos aquí es usar la elaboración de prototipos como una parte del SDLC tradicional. Desde esta perspectiva, la elaboración de prototipos se considera como un método adicional y especializado para determinar los requerimientos de información de los usuarios.

## CÓMO DESARROLLAR UN PROTOTIPO

Los lineamientos de esta sección para desarrollar un prototipo son avanzados. El término *elaboración de prototipos* se interpreta en el sentido de la última definición que se explicó, es decir, un prototipo de características seleccionadas que incluirá algunas pero no todas las características; uno que, si tiene éxito, será parte del sistema final que se entregue.

Como se ilustra en la figura 6.2, la elaboración de prototipos es una excelente forma de obtener retroalimentación sobre el sistema propuesto y sobre la facilidad con que está cumpliendo las necesidades de información de sus usuarios. El primer paso de la elaboración de prototipos es estimar los costos necesarios para la construcción de un módulo del sistema.



**FIGURA 6.2**

La obtención de retroalimentación del usuario da como resultado pantallas mejoradas que satisfacen mejor los requerimientos del usuario.

Si los costos del tiempo de programadores y analistas y los del equipo que utilizarán están dentro del presupuesto, se puede proceder a la elaboración del prototipo. La elaboración de prototipos es una excelente forma de facilitar la integración del sistema de información con el sistema principal de la organización.

## LINEAMIENTOS PARA DESARROLLAR UN PROTOTIPO

Una vez que se ha tomado la decisión de elaborar un prototipo, se deben observar cuatro lineamientos principales al integrar la elaboración de prototipos con la fase de determinación de requerimientos del SDLC:

1. Trabajar en módulos manejables.
2. Construir rápidamente el prototipo.
3. Modificar el prototipo en iteraciones sucesivas.
4. Poner énfasis en la interfaz de usuario.

Como puede ver, los lineamientos sugieren acciones relativas al prototipo que necesariamente se interrelacionan. Cada uno de los lineamientos se explica en las subsecciones siguientes.

**El trabajo en módulos manejables** Cuando el prototipo de algunas de las características de un sistema se integra para formar un modelo funcional, es indispensable que el analista trabaje en módulos manejables. Una ventaja evidente de la elaboración de prototipos es que no es necesario ni deseable construir un sistema operativo completo para los propósitos del prototipo.

Un módulo manejable es aquel que permite a los usuarios interactuar con sus características clave pero que se puede construir de forma separada de otros módulos del sistema. Las características del módulo que se juzgan de menor importancia se omiten intencionalmente en el prototipo inicial.

**Construcción rápida del prototipo** La rapidez es esencial para la elaboración exitosa del prototipo de un sistema de información. Recuerde que una de las quejas expresadas en contra del SDLC tradicional es que el intervalo entre la determinación de requerimientos y la entrega de un sistema completo es demasiado largo para satisfacer eficazmente las cambiantes necesidades del usuario.

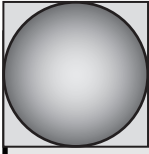
Los analistas pueden usar la elaboración de prototipos con el fin de reducir esta brecha utilizando las técnicas tradicionales de recopilación de información para determinar con precisión los requerimientos de información que surjan sobre la marcha, y a continuación tomar rápidamente las decisiones que den lugar a un modelo funcional. De hecho, el usuario ve y utiliza el sistema muy temprano en el SDLC en lugar de esperar hasta que el sistema se termine para practicar con él.

La preparación de un prototipo operacional, con rapidez y en las etapas tempranas del SDLC, permite al analista comprender mejor cómo desarrollar el resto del proyecto. Al mostrar a los usuarios en las primeras etapas del proceso cómo se ejecutan en la realidad algunas partes del sistema, la elaboración rápida de prototipos evita que se dediquen demasiados recursos a un proyecto que a la larga podría ser imposible de concretar. Más adelante, cuando se explique el RAD, usted verá nuevamente la importancia de la construcción rápida de sistemas.

**Modificación del prototipo** Un tercer lineamiento para desarrollar el prototipo es que su construcción debe soportar modificaciones. Hacer modificable el prototipo significa crearlo en módulos que no sean demasiado interdependientes. Si se observa este lineamiento, se encontrará menos resistencia cuando sea necesario realizar cambios al prototipo.

Generalmente, el prototipo se modifica varias veces al pasar por diversas iteraciones. Los cambios en el prototipo deben propiciar que el sistema se acerque cada vez más a lo que los usuarios consideren importante. Cada modificación necesita otra evaluación por parte de los usuarios.

El prototipo no es un sistema terminado. Abordar la fase de elaboración de prototipos con la idea de que el prototipo requerirá modificaciones es una actitud positiva que demuestra a los usuarios cuán necesaria es su retroalimentación para mejorar el sistema.



### ¿LA ELABORACIÓN DE PROTOTIPOS ES LO MEJOR?

“Como usted sabe, somos un grupo entusiasta. Todavía no somos una dinastía, pero nos estamos esforzando para serlo”, le dice Paul LeGon. Paul (a quien le presentamos en la Oportunidad de consultoría 2.3), con 24 años de edad, es el “rey joven” de Pyramid, Inc., una pequeña empresa editorial independiente pero exitosa que se especializa en libros con cubierta rústica sobre temas poco convencionales. Como analista de sistemas, usted ha sido contratado por Pyramid, Inc., para colaborar en el desarrollo de un sistema de información computarizado para el manejo de la distribución y el inventario del almacén.

“Estamos contratando a muchos trabajadores”, continúa Paul, para convencerlo de la importancia del proyecto de Pyramid. “Y sentimos que Pyramid está perfectamente posicionado en nuestros mercados del norte, el sur, el este y el oeste.”

“Mi ayudante, Ceil Toom, y yo hemos estado trabajando como esclavos, pensando en el nuevo sistema. Y hemos concluido que lo que realmente necesitamos es un prototipo. De hecho, hemos investigado mucho, y cada vez estamos más fascinados con la idea.”

Al tiempo que formula una respuesta para Paul, usted piensa en las pocas semanas que ha trabajado en Pyramid, Inc. Usted cree que los problemas de negocios que su sistema de información debe resolver son muy sencillos. También sabe que las personas de la compañía tienen un presupuesto limitado y no se pueden dar el lujo de gastar como reyes. En realidad, el proyecto entero es bastante pequeño.

Ceil, basándose en lo que Paul ha comentado, dice: “No pretendemos involucrarnos demasiado en esto, pero creemos que la elaboración de prototipos representa la nueva tendencia. Y ahí es donde todos queremos estar. Sabemos que necesitamos un prototipo. ¿Lo hemos convencido?”

Con base en el entusiasmo de Paul y Ceil por la elaboración de prototipos y lo que usted sabe de las necesidades de Pyramid, ¿apoyaría usted la construcción de un prototipo? ¿Por qué sí o por qué no? Redacte una carta en donde explique su decisión y mándesela a Paul LeGon y Ceil Toom. Sustente su decisión argumentando los criterios globales que se deben cumplir para justificar la elaboración de prototipos.

**Énfasis en la interfaz de usuario** La interfaz de usuario con el prototipo (y posteriormente con el sistema) es muy importante. Puesto que en realidad su principal objetivo con el prototipo es conseguir que los usuarios expresen mucho mejor sus requerimientos de información, éstos deben interactuar fácilmente con el prototipo del sistema. Para muchos usuarios la interfaz es el sistema. Esto no debe representar un obstáculo.

Aunque no se desarrollarán muchos aspectos del sistema en el prototipo, la interfaz de usuario se debe desarrollar lo mejor posible para permitir a los usuarios una rápida comprensión del sistema y no sentirse desorientados. Los sistemas interactivos en línea que usan interfaces gráficas son particularmente apropiados para los prototipos. En el capítulo 15 se describen en detalle las consideraciones que son importantes en el diseño de la interfaz de usuario.

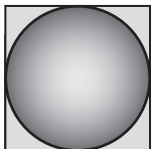
### DESVENTAJAS DE LA ELABORACIÓN DE PROTOTIPOS

Como en cualquier técnica de recopilación de información, la elaboración de prototipos tiene varias desventajas. La primera es que puede ser bastante difícil manejar la elaboración de prototipos como un proyecto en el esfuerzo de sistemas más grandes. La segunda desventaja es que los usuarios y los analistas podrían adoptar un prototipo como si fuera un sistema final cuando de hecho es deficiente y su propósito nunca fue el de servir como sistema terminado.

El analista necesita sopesar estas desventajas contra las ventajas conocidas al decidir si hace el prototipo, cuándo lo hace y de qué partes del sistema lo hace.

### VENTAJAS DE LA ELABORACIÓN DE PROTOTIPOS

La elaboración de prototipos no es necesaria o apropiada en todos los proyectos de sistemas, como hemos visto. Sin embargo, también se deben considerar las ventajas al momento de decidir si se hace el prototipo. Las tres ventajas principales de la elaboración de prototipos son la posibilidad de modificar el sistema en las primeras etapas del desarrollo, la oportunidad de suspender el desarrollo de un sistema que no sea funcional y la posibilidad de desarrollar un sistema que se acerque más a satisfacer las necesidades y expectativas de los usuarios.



## CÓMO ALLANAR EL CAMINO PARA LOS VÍNCULOS DEL CLIENTE

World's Trend (véase en el capítulo 7 una descripción detallada de la corporación) está construyendo un sitio Web para vender mercancía de liquidación que normalmente se vende a través de la Web y por catálogo. En su papel de consultor para la Web recién contratado, Lincoln Cerf se encuentra en una ciudad invernal muy fría, luchando para abrirse paso a través de varias pulgadas de nieve para reunirse con uno de los miembros del equipo de sistemas, Mary Maye, en las oficinas centrales de World's Trend.

Mary da la bienvenida a Lincoln, diciendo: “¡Al menos el clima no parece afectar nuestras ventas en la Web! Siguen adelante sin importar lo que pase”. Lincoln agradece su débil intento por animarlo, sonríe y dice: “Por el correo electrónico que me mandó la semana pasada deduje que está intentando determinar el tipo de información que se debe desplegar en nuestro sitio Web de liquidaciones”.

Mary contesta: “Sí, estoy tratando de organizarlo de la mejor forma posible. Todos nuestros clientes están muy ocupados. Sé que todas las fotografías de nuestra mercancía tardarían mucho tiempo en aparecer en la página si un cliente accede a la Web desde su casa por medio de un módem lento”. Mary continúa: “Linc, en este momento no me preocupa tanto cómo vamos a diseñar nuestro sitio de liquidaciones. En cambio, me preocupa más saber cuánta información necesitamos incluir en una página. Por ejemplo, cuando los artículos están en liquidación, no todos los colores y tallas están disponibles. ¿Qué crees que sea mejor, incluir

alguna información básica y permitir al cliente hacer clic en un botón para pedir más información, o incluir la información más completa posible en una página? Si uso vínculos, entonces podría acomodar más artículos en la pantalla... pero tal vez se vea demasiado orden. A los clientes les agradan las ventas en las cuales el acomodo de la mercancía es un tanto informal”.

Linc abunda sobre el mismo tema, diciendo: “Sí, me pregunto cómo desean los clientes que se organice la información. ¿En realidad has observado cómo se comportan en la Web? Es decir, ¿buscan zapatos cuando compran un traje? Si es así, ¿los zapatos deben aparecer en la página de los trajes o deben vincularse de alguna forma con ésta?”

Mary comenta: “Ésas también son mis dudas. Por ello me pregunto si primero debemos probar este método con la ropa para hombres, antes de ponerlo en práctica con la ropa para mujeres. ¿Qué tal si los métodos de los hombres y las mujeres para comprar en la Web son diferentes?”

Como tercer miembro del grupo de desarrollo del sitio Web de World's Trend, responda mediante un breve informe escrito a Lincoln y Mary si deben usar un prototipo para obtener las recomendaciones de los clientes potenciales sobre el sitio Web propuesto. ¿Qué clase de prototipo es el adecuado? Considere cada clase de prototipo y explique por qué se podría aplicar (o no aplicar) cada uno en este problema. Dedique un párrafo a cada explicación.

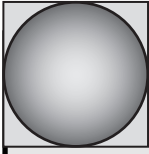
La elaboración exitosa de prototipos depende de una retroalimentación del usuario frecuente y oportuna, lo que sirve para modificar el sistema y hacerlo más receptivo a las necesidades reales. Al igual que cualquier esfuerzo de sistemas, los cambios oportunos son menos costosos que los cambios que se hacen más tarde en el desarrollo del proyecto.

### ELABORACIÓN DE PROTOTIPOS USANDO SOFTWARE COTS

A veces la manera más rápida de elaborar un prototipo es a través de la instalación modular de software COTS (“Comercial Off-The-Shelf”, o software comercial). Aunque el concepto de software COTS se puede entender con facilidad al mirar paquetes conocidos y relativamente económicos como los productos de Microsoft Office, otro software COTS es más complejo y costoso, pero muy útil. Un ejemplo de implementación rápida de software COTS se puede encontrar en la Catholic University, que usa el paquete de software COTS para la planeación de recursos empresariales conocido como PeopleSoft, a través del cual manejan muchas de sus funciones basadas en la Web.

La Catholic University, junto con un importante grupo consultor sobre educación y PeopleSoft, emprendió una implementación rápida y exitosa de un módulo de reclutamiento y admisiones de su software COTS. Ellos iniciaron la implementación en abril de 1999, y para octubre de ese mismo año habían implementado con éxito el reclutamiento y las admisiones para estudiantes de licenciatura. En noviembre del mismo año, implementaron las mismas funciones para estudiantes de postgrado. Otros módulos del software COTS de PeopleSoft que se implementan en la Catholic University incluyen un completo catálogo de cursos en línea, inscripciones en línea y la capacidad para que los estudiantes revisen en línea, desde cualquier parte, sus calificaciones, traslados, facturas y pagos de ayuda financiera.





## EL CRIADERO DE PECES

“Tan sólo tengamos un poco de paciencia. Creo que necesitamos incorporar algunas características más, antes de entregarles el prototipo. De otra manera, este prototipo se hundirá por completo”, dice Sam Monroe, un miembro de su equipo de análisis de sistemas. Los cuatro miembros del equipo están enfrascados en una reunión convocada con precipitación, y discuten acerca del prototipo que desarrollan para un sistema de información que servirá a los gerentes para supervisar y controlar la temperatura del agua, la cantidad de peces puestos a la venta y otros factores en un criadero comercial de peces.”

“Ya tienen bastante que hacer. ¡Vaya!, el sistema empezó con cuatro características y ya llevamos nueve. Siento como si estuviéramos nadando contra la corriente. Ellos no necesitan todo esto. Incluso ni lo quieren”, sostiene Belle Uga, segundo miembro del equipo de análisis de sistemas. “No quiero polemizar, pero opino que tan sólo les demos lo elemental. Ya tenemos suficiente de qué preocuparnos así como están las cosas.”

“Yo creo que Monroe tiene razón”, opina Wally Ide, un tercer miembro del equipo, contraponiéndose un poco a Belle. “Tenemos que darles lo mejor que podamos, aun cuando esto signifique retrasar el prototipo unas cuantas semanas más de lo que prometimos.”

“De acuerdo”, responde Belle con cautela, “pero quiero que ustedes dos expliquen a los gerentes del criadero por qué no les estamos entregando el prototipo. Yo no quiero hacerlo. Y no estoy seguro de que ellos muerdan el anzuelo tan fácilmente”.

Monroe contesta: “Bueno, creo que podemos hacerlo, pero tal vez no consigamos un buen trato al retrasarnos más de lo que queremos. No quiero complicar las cosas”.

Wally coincide: “Es cierto. ¿Por qué mostrar nuestros errores a todos? Además, cuando vean el prototipo, se olvidarán de cualquier queja que tengan. Les encantará”.

Belle saca de su cuaderno un memorando de su última reunión con los gerentes del criadero y lo lee en voz alta. “Agenda de la reunión del 22 de septiembre. ‘Elaboración de prototipos: la importancia del desarrollo rápido, conjuntar al equipo analista de usuarios, obtener una rápida retroalimentación para hacer las modificaciones. . .’” La voz de Belle se desvanece, omitiendo los últimos puntos de la agenda. Después de estos comentarios, Monroe e Ide se miran desconsoladamente.

Monroe habla primero: “Supongo que hicimos el intento de entusiasmar a todos para que esperaran un prototipo, y se involucraran desde el primer día”. Percatándose de que usted ha permanecido en silencio, Monroe continúa: “Pero las aguas aún están agitadas. ¿Qué crees que debemos hacer?”, le pregunta a usted.

En su calidad de cuarto miembro del equipo de análisis de sistemas, ¿qué acciones cree que deban emprenderse? En uno o dos párrafos, envíe un mensaje de correo electrónico a sus compañeros de equipo, con la respuesta a las siguientes preguntas: ¿Deben incorporarse más características al prototipo del sistema del criadero antes de entregarlo a los gerentes para que experimenten con él? ¿Qué tan importante es el desarrollo rápido del prototipo? ¿Cuáles son las ventajas y desventajas de incorporar más características al prototipo o de entregarle al cliente un prototipo más elemental que el que se le prometió? Finalice su mensaje con una recomendación.

## EL PAPEL DEL USUARIO EN LA ELABORACIÓN DE PROTOTIPOS

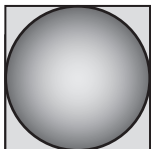
El papel del usuario en la elaboración de prototipos se puede resumir en dos palabras: intervención honrada. Sin la intervención del usuario hay poca razón para elaborar el prototipo. Los comportamientos precisos y necesarios para interactuar con un prototipo pueden variar, pero está claro que el usuario es fundamental en el proceso de la elaboración de prototipos. Comprendiendo la importancia que tiene el usuario en el éxito del proceso, los miembros del equipo de análisis de sistemas deben propiciar y recibir de buena manera la retroalimentación y deben evitar su propia resistencia natural a cambiar el prototipo.

### INTERACCIÓN CON EL PROTOTIPO

Hay tres formas principales en las que un usuario puede ayudar en la elaboración de prototipos:

1. Experimentando con el prototipo.
2. Dando reacciones sinceras sobre el prototipo.
3. Sugiriendo adiciones o eliminaciones al prototipo.

Los usuarios deben tener libertad para experimentar con el prototipo. En contraste con una simple lista de características de sistemas, el prototipo permite a los usuarios la interacción real. Una forma de facilitar esta interacción es instalar un prototipo en un sitio Web interactivo.



## ESTE PROTOTIPO ESTÁ TODO MOJADO

“Puede cambiarse. Recuerde que no es un producto final”, afirma Sandy Beach, analista de sistemas de RainFall, un fabricante de bañeras de fibra de vidrio y cancelas para baño. Con ansiedad, Beach intenta tranquilizar a Will Lather, un planificador de producción de Rainfall que examina cuidadosamente el primer informe impreso con el prototipo del nuevo sistema de información.

“Bueno, está bien”, dice Lather tranquilamente. “No quisiera darle ninguna molestia. Veamos. . . sí, *aquí* están”, dice cuando al fin localiza el informe mensual que resume las materias primas adquiridas, las utilizadas y las que están en inventario.

Lather continúa hojeando el abultado informe impreso por la computadora. “Esto estará bien.” Deteniéndose en un informe, comenta: “Le entregaré una copia de esta parte a la señorita Fawcett para la gente de Contabilidad”. Hojeando otras cuantas páginas, dice: “Y el encargado de Aseguramiento de la Calidad realmente tiene que ver esta columna de cifras, aunque el resto no sea de mucho interés para él. La circularé y sacaré una copia para él. Quizá también deba informar por teléfono parte de esto al almacén”.

Cuando Sandy se prepara para salir, Lather levanta las páginas del informe y comenta: “El nuevo sistema será de gran ayuda. Me aseguraré de que todos lo conozcan. En fin, cualquier cosa es mejor que ‘el viejo monstruo’. Me alegro de que tengamos algo nuevo”.

Sandy abandona la oficina de Lather sintiéndose confundido. Pensando en su reciente conversación, empieza a preguntarse por qué Contabilidad, Aseguramiento de la Calidad y el almacén no están recibiendo lo que Will cree que deberían. Beach se comunica por teléfono con algunas personas y

confirma que lo dicho por Lather es verdad. Ellos necesitan los informes y no les están llegando.

Avanzada la semana, Sandy se acerca a Lather y le pide su opinión sobre redirigir la salida del sistema al igual que cambiar algunas de las características del mismo. Estas modificaciones permitirían a Lather obtener respuestas en pantalla relativas a escenarios del tipo “qué pasaría si” en relación con cambios en los precios que cobran los proveedores o con cambios en el porcentaje de calidad de las materias primas que entregan los proveedores (o ambos), además de permitirle ver lo que ocurriría si un embarque se retrasara.

Lather se molesta visiblemente con las sugerencias de Sandy acerca de alterar el prototipo y su salida. “Oh, no lo haga por mi causa. En realidad así está bien. No me molesta la responsabilidad de hacer llegar la información a la gente. De cualquier manera, siempre los atiborro de material. Créame, el prototipo funciona bastante bien. No me gustaría nada que nos lo quitara en este momento. Dejémoslo como está.”

Sandy está contento de que Lather se muestre tan satisfecho con la salida del prototipo, pero le preocupa la reticencia de éste a cambiar el prototipo puesto que él siempre estuvo exhortando a los usuarios a que lo consideraran como un producto cambiante, no uno terminado.

Escriba un breve informe a Sandy en el cual mencione los cambios al prototipo motivados por las reacciones de Will. En un párrafo, explique formas en que Sandy puede calmar los temores de Lather sobre “quitarles” el prototipo. Discuta en un párrafo algunas acciones que pueden tomarse para alertar a los usuarios sobre la naturaleza *cambiante* de un prototipo antes de que éste se ponga a prueba.

Otro aspecto del papel de los usuarios en la elaboración de prototipos requiere que proporcionen reacciones sinceras acerca del prototipo. Por desgracia, estas reacciones no se dan bajo demanda. Más bien, lograr que los usuarios se sientan suficientemente seguros para dar una reacción sincera es parte de la relación entre analistas y usuarios que su equipo debe procurar establecer.

Los analistas necesitan estar presentes por lo menos en el momento en que ocurre la experimentación. Entonces pueden observar las interacciones de los usuarios con el sistema, y están obligados a ver las interacciones que nunca planearon. En la figura 6.3 se muestra un formulario contestado para observar la experimentación del usuario con el prototipo. Algunas de las variables que debe observar incluyen las reacciones del usuario al prototipo, las sugerencias del usuario para cambiar o ampliar el prototipo, las innovaciones del usuario para usar el sistema de formas completamente nuevas y cualquier revisión planeada para el prototipo que le ayuda a establecer las prioridades.

Un tercer aspecto del papel de los usuarios en la elaboración de prototipos es su disposición para sugerir adiciones o eliminaciones de las características experimentadas. El papel del analista es producir tales sugerencias asegurando a los usuarios que la retroalimentación que ellos proporcionan se toma en serio, observándolos cuando interactúen con el sistema y haciendo entrevistas cortas y específicas con usuarios que tengan experiencias relacionadas con el prototipo. Aunque se les pedirá a los usuarios que den sugerencias e innovaciones para el prototipo, al final el analista tiene la responsabilidad de analizar esta retroalimentación y traducirla a los cambios que sean necesarios. Para facilitar el proceso de la elaboración de prototipos, el analista debe comunicar claramente a los usuarios los propósitos de dicha elaboración, junto con la idea de que sólo es valiosa cuando los usuarios se involucran significativamente.

**FIGURA 6.3**

Un paso importante de la elaboración de prototipos es registrar adecuadamente las reacciones del usuario, sus sugerencias e innovaciones, así como los planes de revisión.

Formulario de evaluación del prototipo				
Nombre del observador		Fecha		
Chip Puller		1/06/2003		
Nombre del sistema o del proyecto		Empresa o ubicación		
Sistema de microcomputadora		Central Pacific University		
Nombre o número del programa		Versión		
Mtto. preventivo		1		
Nombre del usuario	Usuario 1	Usuario 2	Usuario 3	Usuario 4
Mike C.	Dot M.			
Periodo de observación	1/06/2003 A.M.	1/06/2003 A.M.		
Reacciones del usuario	Generalmente favorables, emocionado con el proyecto.	¡Excelente!		
Sugerencias del usuario	Agregar la fecha del día en que se ejecutó el mantenimiento.	Colocar el número de formulario en la parte superior para referencia. Poner la palabra SEMANAL en el título.		
Innovaciones				
Planes de revisiones	Modificar el 1/08/2003. Revisar con Dot y Mike.			

## DESARROLLO RÁPIDO DE APLICACIONES

El desarrollo rápido de aplicaciones (RAD) es un enfoque orientado a objetos para el desarrollo de sistemas que incluye un método de desarrollo así como también herramientas de software. Es lógico discutir RAD y la elaboración de prototipos en el mismo capítulo, debido a que están conceptualmente muy unidos. Ambos tienen como meta la reducción del tiempo que generalmente se necesita en un SDLC tradicional entre el diseño y la implementación del sistema de información. Finalmente, el RAD y la elaboración de prototipos se enfocan en satisfacer más de cerca los requerimientos cambiantes de los negocios. Una vez que ha aprendido los conceptos de la elaboración de prototipos, es mucho más fácil entender la esencia del RAD, que se puede considerar como una implementación específica de la elaboración de prototipos.

Algunos desarrolladores están considerando al RAD como un enfoque útil para los nuevos entornos de comercio electrónico basados en la Web, en el cual podría ser importante el estatus de primero en tomar la iniciativa de un negocio. En otras palabras, para poner una aplicación en la Web antes que sus competidores, las empresas podrían requerir que su equipo de desarrollo experimente con el RAD.

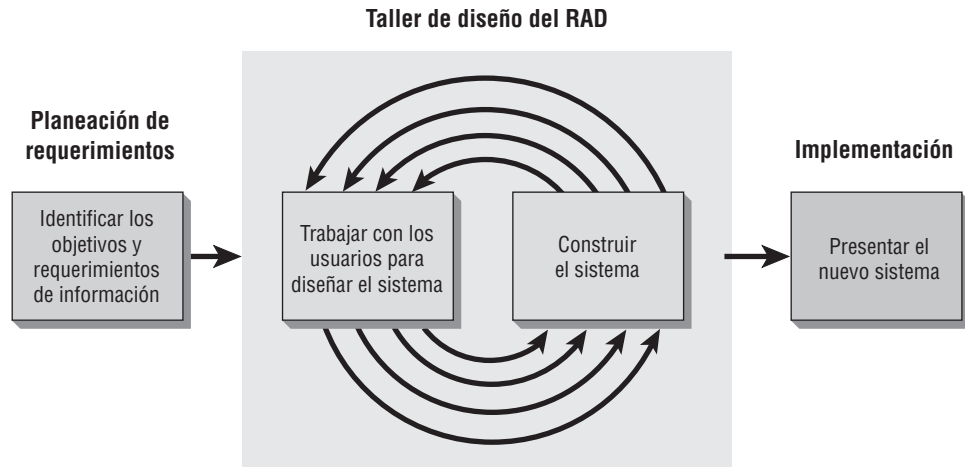
### FASES DEL RAD

Hay tres fases amplias del RAD que vinculan a usuarios y analistas en la evaluación, diseño e implementación. La figura 6.4 describe estas fases. Observe que el RAD involucra a los usuarios en cada parte del esfuerzo de desarrollo, con una intensa participación en la parte de negocios del diseño.

**Fase de planeación de requerimientos** En esta fase, usuarios y analistas se reúnen para identificar los objetivos de la aplicación o sistema y para identificar los requerimientos de

**FIGURA 6.4**

El taller de diseño del RAD es el corazón del proceso interactivo de desarrollo.



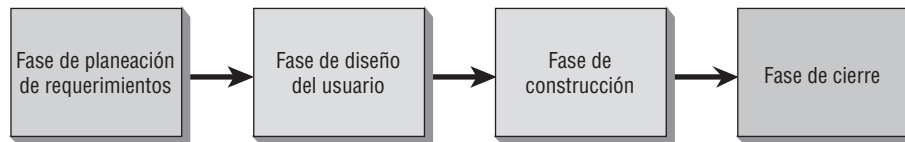
información que surgen de dichos objetivos. Esta fase requiere que ambos grupos se involucren intensamente; no se trata simplemente de firmar una propuesta o documento. Además, esto podría involucrar a usuarios de los diferentes niveles de la organización (como se trató en el capítulo 2). En la fase de planeación de requerimientos, cuando aún se están determinando los requerimientos de información, usted podría estar trabajando con el director de información (si es una organización grande) así como también con la gente de planeación estratégica, sobre todo si usted está trabajando con una aplicación de comercio electrónico cuyo propósito es impulsar los objetivos estratégicos de la organización. La orientación en esta fase tiene el objetivo de resolver los problemas de negocios. Aunque algunas de las soluciones propuestas podrían surgir de la tecnología de información disponible, el enfoque siempre será alcanzar los objetivos del negocio.

**Taller de diseño del RAD** El proceso de diseñar y refinar los prototipos se puede representar mejor como un taller. Cuando imagina un taller, sabe que la participación es intensa, no pasiva, y que generalmente se hace con las manos. Normalmente los usuarios están sentados en mesas redondas o en una configuración en forma de U de sillas con escritorios adheridos donde cada persona puede ver a otra y donde hay espacio para trabajar con una computadora portátil. Si usted es bastante afortunado para disponer de un salón para sistemas de apoyo a la toma de decisiones en grupo (GDSS) en la compañía o a través de una universidad local, utilícelo para conducir por lo menos una parte de su taller de diseño de RAD.

Durante el taller de diseño del RAD, los usuarios responden a los prototipos operativos reales y los analistas refinan los módulos diseñados (utilizando algunas de las herramientas de software que se mencionan más adelante) basados en las respuestas del usuario. El formato del taller es muy emocionante y estimulante, y si están presentes los usuarios y los analistas experimentados, no hay ninguna duda de que este esfuerzo creativo puede impulsar el desarrollo a gran velocidad.

**Fase de implementación** En la figura anterior, puede ver que los analistas están trabajando intensamente con los usuarios durante el taller para diseñar los aspectos del negocio o no técnicos del sistema. Tan pronto como sean convenidos estos aspectos y los sistemas sean construidos y se refinan, los nuevos sistemas, o parte de ellos, son probados e introducidos en la organización. Debido a que el RAD se puede usar para crear las nuevas aplicaciones de comercio electrónico para las cuales no hay ningún sistema viejo, por lo general no se necesita ejecutar los sistemas viejos y nuevos en paralelo antes de la implementación (además que no hay forma real de hacerlo).

En este punto, el taller de diseño del RAD habrá generado el interés, sentido de pertenencia del usuario y la aceptación de la nueva aplicación. Generalmente, el cambio que se produce de esta forma es mucho menos doloroso que cuando un sistema se entrega con poca o ninguna participación del usuario.



**FIGURA 6.5**  
Fases del RAD de Martin.

**Enfoques pioneros de Martin para el RAD** En la figura 6.5 usted puede ver nuestra conceptualización de las fases originales del RAD de James Martin. En la primera fase Martin explica la planeación de requerimientos. Aquí los usuarios de alto nivel deciden qué funciones debe incluir la aplicación.

En la segunda fase, llamada fase de diseño del usuario, Martin caracteriza a los usuarios como ocupados en discutir los aspectos no técnicos del diseño del sistema, con la ayuda de los analistas. La fase del taller de diseño del RAD incorpora las fases del usuario y la de construcción en una, debido a que la naturaleza muy interactiva y visual del proceso de diseño y refinación están ocurriendo de una forma interactiva y participativa.

En la fase de construcción se realizan muchas actividades diferentes. Cualesquier diseños que se creen en la fase anterior se mejoran más con las herramientas del RAD. Tan pronto como las nuevas funciones están disponibles, se muestran a los usuarios para la interacción, comentarios y revisión. Con las herramientas del RAD, los analistas pueden hacer cambios continuos en el diseño de las aplicaciones.

En la cuarta y última fase de Martin, la fase de cierre, la aplicación recientemente desarrollada reemplazará a la anterior. Mientras está ejecutándose en paralelo con la aplicación anterior, la nueva se prueba, los usuarios son entrenados y los procedimientos de la organización se cambian antes de que ocurra el cierre.

**Herramientas de software para el RAD** Como usted podría esperar, por lo regular las herramientas de software para el RAD son las más nuevas, con frecuencia orientadas a objetos. Algunos ejemplos son programas muy conocidos tales como Microsoft Access, Microsoft Visual Basic, Visual C++ y Microsoft .NET. (Véase el capítulo 18 para una explicación más detallada del enfoque orientado a objetos.)

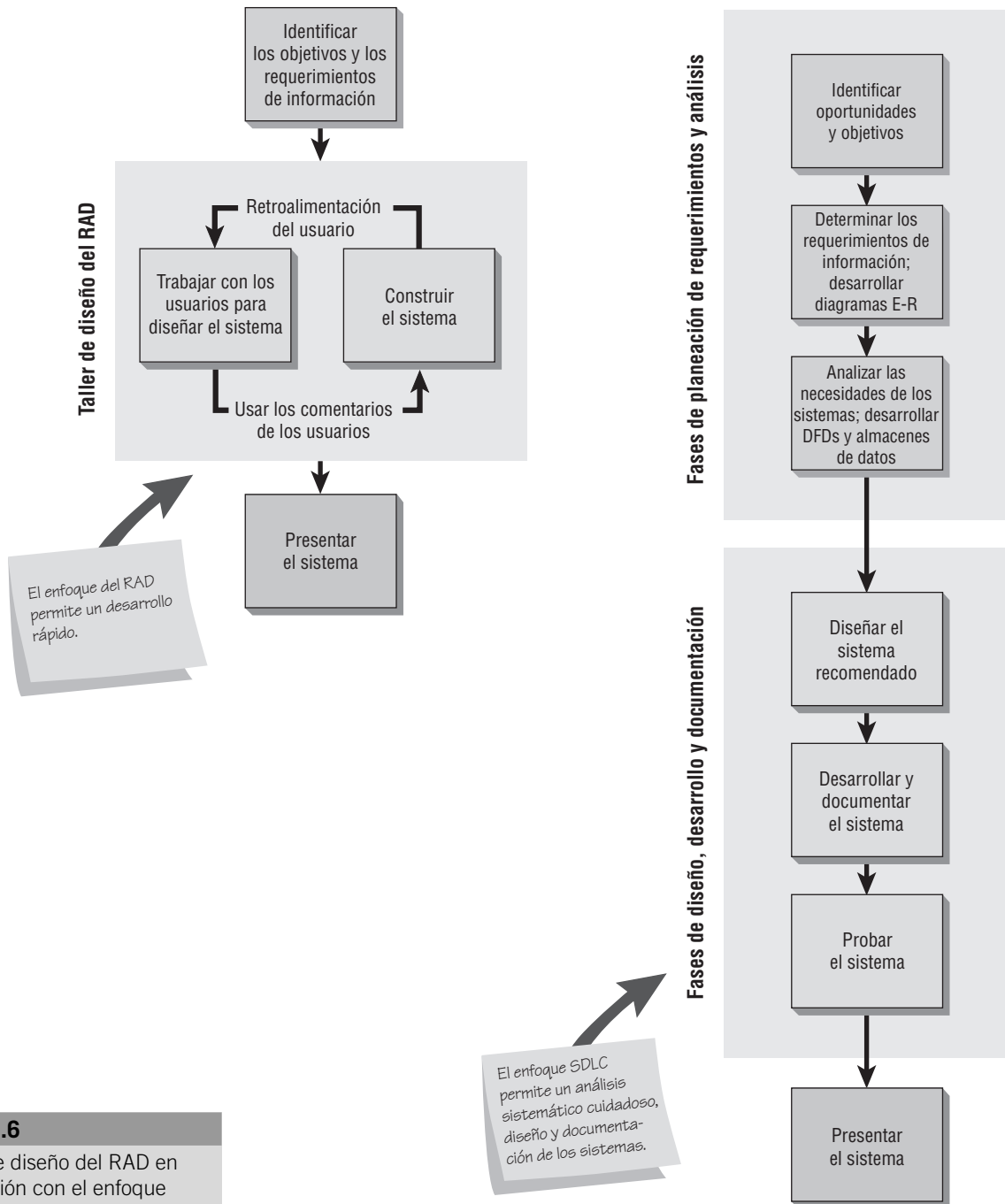
Una forma en que las herramientas difieren entre sí está en sus capacidades para dar soporte a las aplicaciones cliente/servidor (por ejemplo, MS Access no da soporte, Visual Basic sí lo da) así como también su facilidad de uso y el nivel de conocimientos de programación que se requieren. La mayoría de las aplicaciones del RAD se usan para aplicaciones pequeñas basadas en PC, aunque su verdadero poder podría radicar en las aplicaciones cliente/servidor que necesitan ejecutarse a través de múltiples plataformas.

Aunque hay identificadas casi tantas fases diferentes del RAD así como hay analistas, las cuatro fases propuestas por Martin —planeación de requerimientos, diseño del usuario, la construcción y cierre— son útiles. Examinemos cada una con más detalle, comparándolas y contrastándolas con las características de la elaboración de prototipos clásica y el SDLC tradicional.

## RAD EN COMPARACIÓN CON EL SDLC

En la figura 6.6 puede comparar las fases del SDLC con aquellas detalladas para el RAD al principio de esta sección. Observe que el principal propósito del RAD es acortar el SDLC y de esta forma responder más rápidamente a los requerimientos de información dinámicos de las organizaciones. El SDLC toma un enfoque más metódico y sistemático que asegura la integridad y exactitud y tiene como propósito la creación de sistemas que se integran bien en los procedimientos estándar de negocio y en la cultura.

La fase del taller de diseño del RAD difiere de las fases de diseño estándar del SDLC, debido a que las herramientas de software del RAD se usan para generar pantallas y exhibir



**FIGURA 6.6**

El taller de diseño del RAD en comparación con el enfoque del SDLC.

el flujo global del funcionamiento de la aplicación. Así, cuando los usuarios aprueban este diseño, están conviniendo en una representación del modelo visual, no sólo en un diseño conceptual representado en papel, como tradicionalmente se hace.

La fase de implementación del RAD es, en muchas formas, menos estresante que otras, debido a que los usuarios han ayudado a diseñar los aspectos de negocios del sistema y saben perfectamente qué cambios se harán. Hay pocas sorpresas, y el cambio es algo a lo que se le da la bienvenida. Con frecuencia, cuando se utiliza el SDLC y los analistas están separados de los usuarios, hay mucho tiempo entre el desarrollo y el diseño. Durante este periodo, los requerimientos pueden cambiar y los usuarios se pueden sorprender si el producto final es diferente del que se anticipó durante muchos meses.

**Cuándo utilizar el RAD** En su función de analista, necesita aprender tantos enfoques y herramientas como sea posible que lo ayuden a hacer mejor su trabajo. Ciertas aplicaciones y trabajo de sistemas darán lugar a ciertas metodologías. Considere utilizar RAD cuando:

1. su equipo incluya a programadores y analistas que tengan experiencia con él, y
2. haya razones de negocios urgentes para acelerar una parte del desarrollo de la aplicación; o
3. cuando esté trabajando con una nueva aplicación de comercio electrónico y su equipo de desarrollo crea que el negocio puede beneficiarse ampliamente sobre sus competidores siendo innovador si esta aplicación está entre las primeras en aparecer en la Web; o
4. cuando los usuarios sean maduros y estén altamente comprometidos con las metas organizacionales.

**Desventajas del RAD** Las dificultades con el RAD, como con otras clases de elaboración de prototipos, se originan debido a que los analistas de sistemas intentan apresurar demasiado el proyecto. Suponga que se contratan dos carpinteros para construir dos cobertizos de almacenamiento para dos vecinos. El primer carpintero sigue la filosofía de SDLC, mientras que el segundo la del RAD.

El primer carpintero es sistemático, cataloga cada herramienta, cada podadora y cada uno de los muebles del patio para determinar el tamaño correcto del cobertizo, diseña un plano del cobertizo y anota las especificaciones para cada parte de madera y hardware. El carpintero construye el cobertizo con poca pérdida y tiene la documentación precisa sobre cómo fue construido el cobertizo por si cualquiera quisiera construir otro parecido, repararlo o pintarlo del mismo color.

El segundo carpintero va directo al proyecto y calcula el tamaño del cobertizo, consigue un camión de madera y hardware, construye una estructura y discute con el dueño de la propiedad las modificaciones necesarias si no están disponibles ciertos materiales y hace un viaje para devolver la madera que no se usa. El cobertizo se construye rápidamente, pero si no se hace un plano, nunca existe la documentación.

## PROGRAMACIÓN EXTREMA

La programación extrema (XP) es un enfoque de desarrollo de software (tratado en el capítulo 3) que adopta lo que generalmente designamos como prácticas de desarrollo de software aceptable y las lleva al extremo. Por ejemplo, la retroalimentación es importante para los programadores, analistas, diseñadores, usuarios y computadoras (como verá en el capítulo 14). Así que la programación extrema usa ciclos de retroalimentación cada vez más rápidos e intensos, que proporcionan más información.

La administración de proyectos es importante (como se vio en el capítulo 3), de tal manera que la programación extrema intenta definir rápidamente un plan global del sistema, desarrollar y liberar rápidamente el software y posteriormente revisarlo continuamente para incorporarle características adicionales. Los programadores, analistas y diseñadores ordinarios que trabajan independientemente y luego integran su trabajo logran resultados sólidos; los programadores extremos que trabajan en pareja pueden ser excelentes. Pero la programación extrema no sólo se basa en los resultados. Se basa en los valores, principios y prácticas. Ahora examinaremos cómo los valores y principios de XP dan forma al desarrollo de sistemas extremos.

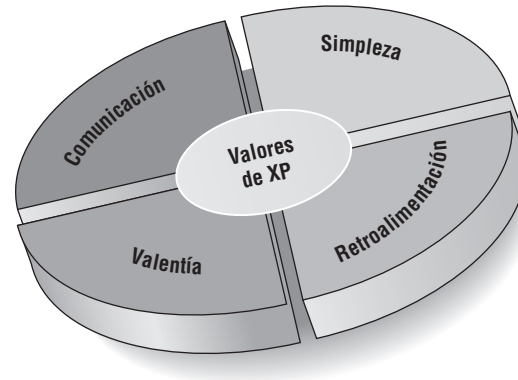
## VALORES Y PRINCIPIOS DE LA PROGRAMACIÓN EXTREMA

Para la programación extrema es importante que se declaren los valores y principios que crean el contexto para la colaboración entre programadores y clientes. Para considerarse



**FIGURA 6.7**

Los valores son de gran importancia para la programación extrema.



analista de XP, se debe apegar a los siguientes valores y principios desarrollados por Beck (2000).

**Cuatro valores de XP** Hay cuatro valores que crean un entorno en el cual se pueden servir adecuadamente diseñadores y negocios. Debido a que con frecuencia hay una tensión entre lo que los diseñadores hacen a corto plazo y lo que es comercialmente deseable a largo plazo, es importante que esté consciente de apoyar valores que formarán una base para colaborar juntos en un proyecto de software. Como se muestra en la figura 6.7, los cuatro valores son comunicación, sencillez, retroalimentación y valentía.

Empecemos con la comunicación. Cada esfuerzo humano tiene la posibilidad de fallar en la comunicación. Los proyectos de los sistemas que requieren una actualización constante y un diseño técnico son especialmente propensos a dichos errores. Agregue a este proyecto fechas límite ajustadas, jerga especializada y el estereotipo de que los programadores prefieren hablar con las máquinas que con las personas, y usted tiene los ingredientes para algunos problemas serios de comunicación. Los proyectos pueden ser retrasados; se puede resolver el problema equivocado; se castiga a los programadores incluso por mencionar a los gerentes que hay problemas; las personas abandonan o se unen al proyecto a la mitad sin estar al corriente; y así continúa la letanía.

Prácticas típicas de XP tal como la programación en parejas (colaboración de dos programadores, descrita más adelante en el capítulo), estimación de las tareas y las pruebas del software, requieren de una buena comunicación. Los problemas se resuelven rápidamente, los agujeros se tapan y la opinión débil se fortalece rápidamente a través de la interacción con otros en el equipo. Un instructor de XP, como se describió en el capítulo 3, está presente para observar si alguien ha interrumpido la comunicación y para reunirlos.

El segundo valor de la programación extrema es la simpleza. Cuando estamos trabajando en un proyecto de desarrollo de software, nuestra primera reacción es abrumarnos por la complejidad y magnitud de la tarea. Sin embargo, usted no puede correr si no sabe caminar, ni caminar si no sabe ponerse de pie. La simpleza en el desarrollo de software significa que empezaremos con la cosa más sencilla que podemos hacer.

La simpleza lleva tiempo, y es algo en lo que el instructor de XP podría ayudarle. El valor de XP de simpleza nos pide que hoy hagamos la cosa más sencilla, comprendiendo que mañana se podría cambiar un poco. Esto requiere un enfoque claro de las metas del proyecto y realmente es un valor básico.

La retroalimentación es el tercer valor básico que es importante para tener un enfoque de la programación extrema. Cuando piensa en la retroalimentación en este contexto, es bueno considerar que ésta se relaciona con el concepto de tiempo. Una retroalimentación buena y concreta, que es útil para el programador, analista y cliente puede ocurrir en segundos, minutos, días, semanas o meses, dependiendo de lo que se necesita, quién está comunicando y lo que se hará con dicha retroalimentación. Un colega programador podría encontrar un caso de prueba que hiciera que un código que usted escribió fallara. Esto



podría ocurrir sólo horas antes de la fecha de entrega, pero dicha retroalimentación es casi invaluable en lo que se refiere a poder cambiar lo que no está funcionando antes de que se acepte y se inserte en el sistema.

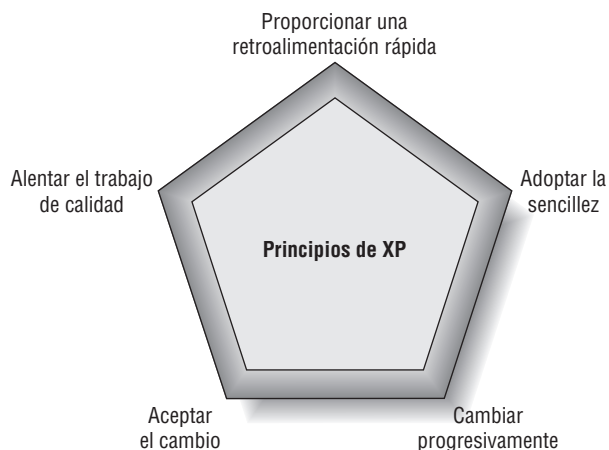
La retroalimentación ocurre cuando los clientes crean pruebas funcionales para todas las historias que los programadores habrán de implementar. (Véase más adelante en este capítulo las historias del usuario.) La retroalimentación crítica sobre el programa de trabajo viene de clientes que comparan la meta del plan con el progreso que se ha tenido. La retroalimentación ayuda a los programadores a hacer los ajustes y permite a los negocios tener una experiencia a tiempo de lo que el nuevo sistema se parecerá una vez que sea totalmente funcional.

La valentía es el cuarto valor enunciado en la programación extrema. La valentía tiene que ver con un nivel de confianza que debe existir en el equipo de desarrollo. Significa que no se debe tener miedo de tirar una tarde o un día de programación y empezar de nuevo si todo está mal. Significa tomar en cuenta los propios instintos (y resultados de la prueba) respecto de lo que funciona y lo que no.

La valentía también significa responder a una retroalimentación concreta, tomar una decisión basándose en el presentimiento de un compañero de equipo cuando creen que tienen una forma más simple o mejor de lograr su meta. La valentía es un valor de alto riesgo y de alta recompensa que anima a la experimentación que el equipo puede tomar de una forma más rápida e innovadora para lograr su meta. La valentía significa que usted y sus compañeros se tienen suficiente confianza mutua y en sus clientes como para actuar en formas que mejorarán continuamente lo que se está haciendo en el proyecto, aun cuando ello requiera tirar el código, reconsiderar las soluciones o, más aún, simplificar los enfoques. La valentía también implica que usted, como analista de sistemas, aplique con empeño las prácticas extremas de XP.

**Principios básicos de XP** En un mundo perfecto, los clientes y su equipo de desarrollo de software estarían de acuerdo en todo y no sería necesaria la comunicación. Sabemos que no existe el mundo ideal. ¿Pero cómo podemos hacer nuestros proyectos de desarrollo de software más parecidos al ideal? Una razón para que esto no ocurra se debe a que hasta ahora estamos intentando operar en un sistema poco claro de valores compartidos. Son un buen principio, pero realmente no son operativos en el punto en que podamos medir nuestro éxito de forma significativa. De manera que trabajamos para derivar los principios básicos que nos pueden ayudar a verificar si lo que estamos haciendo en nuestro proyecto de software realmente está midiendo los valores que compartimos.

Aunque hay alrededor de una docena de principios que podemos derivar de nuestros valores, los principios básicos que describimos son: proporcionar una rápida retroalimentación, dar por hecho la sencillez, cambiar progresivamente, aceptar el cambio y alentar el trabajo de calidad. En la figura 6.8 se ilustran dichos principios.



**FIGURA 6.8**

Cinco principios de XP guían al analista de sistemas a través de un proyecto de XP exitoso.

El principio básico para recordar respecto a la retroalimentación rápida es que para que las personas o los sistemas hagan una conexión entre estímulo y reacción, la retroalimentación debe ocurrir en un intervalo razonablemente corto. Si a una impresora se le termina el papel, debe desplegar instantáneamente un mensaje “no hay papel” como retroalimentación para el usuario, de manera que la situación se pueda remediar y la impresión pueda continuar. La retroalimentación rápida para el equipo de desarrollo significa que entre más cercano sea el tiempo de una acción (codificar una característica derivada de un reporte del usuario) al tiempo de la comprobación, más significativa será la retroalimentación (los resultados de la prueba). Entre más pronto en la vida de un sistema éste se ponga en producción (en lugar de simplemente estar en desarrollo), mayor será el valor de la retroalimentación para el negocio al medir si el sistema está cumpliendo sus metas.

El siguiente principio básico es que el equipo de desarrollo debe adoptar la simpleza. La premisa es que alrededor de 90 por ciento de los problemas se puede resolver con absoluta sencillez. Observe que esto se contrapone a la mayoría del entrenamiento tradicional, el cual les pide a los desarrolladores que planeen para el futuro, entiendan todas las interfaces, y así sucesivamente, antes de empezar. La programación extrema dice que la simpleza rige el día, y que los programadores deben confiar en su habilidad de agregar la complejidad el próximo día si se requiere. Éste es un principio muy difícil de dominar para muchos diseñadores.

El tercer principio que examinamos es el cambio progresivo. Esto significa que usted constantemente está haciendo el cambio más pequeño posible que aún resulte en una diferencia en el esfuerzo de desarrollo. Ningún cambio mayor en el código, el equipo y los requerimientos del negocio. Ellos requieren cambio progresivamente, incluso después de que se libera el producto. Esto se adapta bien con la idea de XP de evolución.

Un cuarto principio básico que podemos derivar de los valores de XP es el de aceptar el cambio. Necesitamos mantener abiertas todas nuestras opciones, pero, al mismo tiempo, necesitamos ser capaces de resolver cualquier obstáculo que se presente. Aunque siempre hay pros y contras, sabemos con seguridad que el cambio es bienvenido. Ese dinamismo permite que el proyecto siga adelante y anima el espíritu del equipo del proyecto. El cambio es bueno.

El último principio es la noción de alentar el trabajo de calidad. El principio proviene de la idea de que todos los participantes deben hacer un trabajo de calidad. Si no hicieran trabajo de calidad, ¿por qué considerar ser incluidos en un esfuerzo de programación extrema? El punto es hacer el trabajo agradable, trabajar adecuadamente con el equipo y mantener el proyecto sano y salvo.

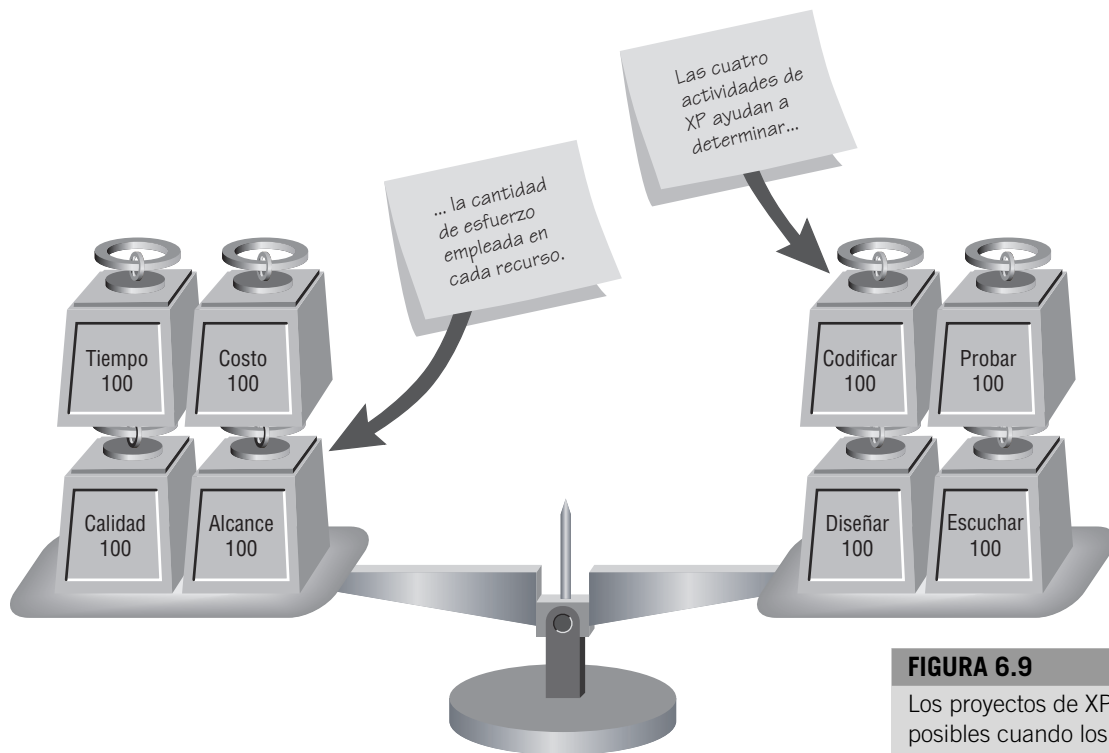
Existen algunos principios más que ayudarán a los desarrolladores a saber cómo proceder cuando surja una situación particular. En pocas palabras: incluyen la obligación de enseñar a aprender; el estímulo para hacer una pequeña inversión inicial de manera que se haga un buen, pero no extravagante, trabajo; jugar para ganar, no jugar para evitar perder; y usar experimentos concretos para probar el trabajo que se está haciendo.

Otros conceptos importantes que apoyan la programación extrema son la idea de usar la comunicación abierta y honrada sin miedo; aprovechar las tendencias naturales de las personas (querer ser exitoso, interactuar con otros, tener la autonomía en su trabajo, ser parte de un equipo ganador, ser confiado, tener en funcionamiento su software); asumir la responsabilidad de hacer algo en lugar de ordenarle a alguien que lo haga; adaptar localmente el enfoque que está aprendiendo para la programación extrema; y buscar utilizar una medida honrada que no pretenda alcanzar una precisión que no existe.

## ACTIVIDADES, RECURSOS Y PRÁCTICAS DE LA PROGRAMACIÓN EXTREMA

La programación extrema involucra varias actividades que se necesita completar en algún momento durante el proceso de desarrollo de XP. Esta sección discute dichas actividades, recursos y prácticas que son únicos para la programación extrema.

**Cuatro actividades básicas de XP** Hay cuatro actividades básicas de desarrollo que utiliza la programación extrema. Dichas actividades son codificar, probar, escuchar y diseñar. El analista de XP necesita identificar la cantidad de esfuerzo que entrará en cada actividad y



**FIGURA 6.9**

Los proyectos de XP exitosos son posibles cuando los recursos se utilizan para equilibrar las actividades que se necesitan completar.

el equilibrio necesario con los recursos para completar el proyecto. En la figura 6.9 se muestra dicho equilibrio, en la cual se describe una balanza en donde se pone una serie de pesos. En XP, los pesos del lado derecho son las actividades. Los pesos en el lado izquierdo son los recursos, que se trataron con mayor detalle en el capítulo 3.

Codificar se designa como una actividad dado que no es posible hacer nada sin ella. Un autor afirma que la cosa más valiosa que recibimos de la codificación es el “aprendizaje”. El proceso básicamente es esto: tenga un pensamiento, codifíquelo, pruébelo y vea si ese pensamiento era lógico. También se puede codificar para comunicar ideas que de otra manera permanecerían borrosas o sin forma. Cuando veo su código, puedo generar un nuevo pensamiento. El código fuente es la base para que un sistema sobreviva. Es esencial para el desarrollo.

La segunda actividad básica del desarrollo es probar. La programación extrema da mucha importancia a las pruebas automatizadas. La programación extrema apoya la generación de pruebas escritas para verificar la codificación, la funcionalidad, el rendimiento y la conformidad con los objetivos. XP se apoya en las pruebas automatizadas y existen grandes bibliotecas de pruebas para la mayoría de los lenguajes de programación. Estas pruebas necesitan ser actualizadas conforme sea necesario durante el progreso del proyecto.

Hay razones de corto y largo plazos para hacer pruebas. Probar a corto plazo le proporciona la confianza extrema en lo que está construyendo. Si las pruebas se ejecutan perfectamente usted puede seguir adelante con la confianza renovada. A largo plazo, probar mantiene vivo un sistema, y le permite hacer muchos más cambios de los que serían posibles si ninguna prueba fuera escrita o ejecutada.

La tercera actividad básica del desarrollo es escuchar. En el capítulo 4 aprendimos la importancia de escuchar durante las entrevistas. En la programación extrema, esta actividad se lleva al extremo. Los desarrolladores escuchan de manera activa a sus compañeros de programación. En XP se depende menos de la comunicación formal escrita y por ello escuchar se vuelve una habilidad muy importante.

El desarrollador también escucha de manera activa al cliente. Los desarrolladores asumen que no saben nada acerca del negocio en el que están colaborando, y por lo tanto deben escuchar cuidadosamente a los usuarios para obtener las respuestas a sus preguntas. El desarrollador

necesita entender la eficacia de escuchar. Si no escucha, no sabrá lo que debe codificar o lo que debe probar.

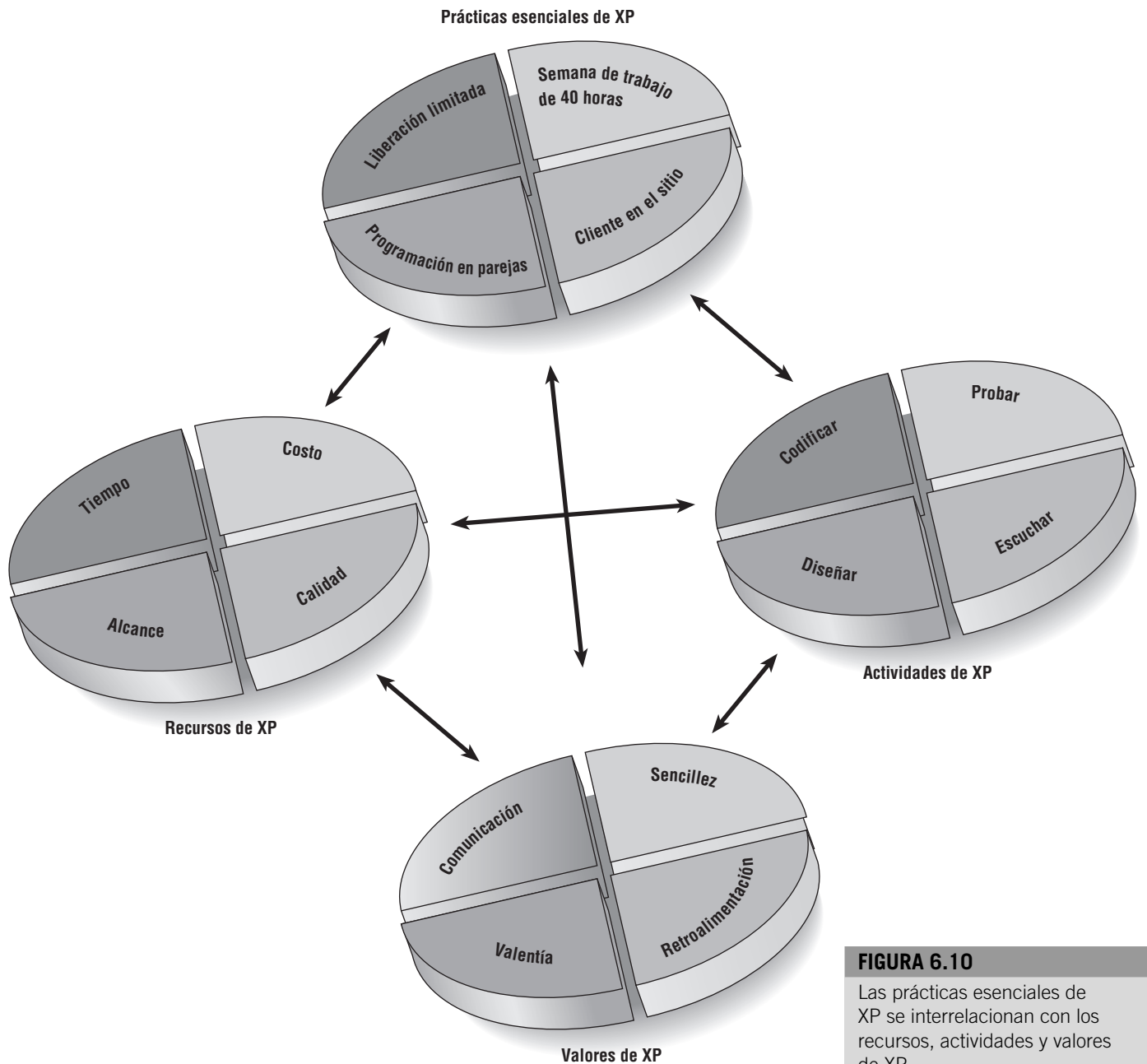
La cuarta actividad básica en el desarrollo es diseñar, lo cual es una forma de crear una estructura para organizar toda la lógica en el sistema. Diseñar es una actividad evolutiva, y por ello los sistemas que se diseñan con un enfoque de la programación extrema se conceptualizan como en constante evolución, siempre diseñándose.

Con frecuencia un buen diseño es simple. También, éste debe permitir la flexibilidad. Diseñar bien permite agregar extensiones al sistema haciendo cambios en un solo lugar. El diseño eficaz ubica la lógica cerca de los datos en que estará operando. Sobre todo, el diseño debe ser útil para todos aquellos que lo necesitarán conforme avance el esfuerzo de desarrollo, incluyendo a clientes y programadores.

**Cuatro variables de control de recursos de XP** Con el objetivo de lograr las actividades descritas anteriormente, los analistas de XP necesitan recursos. Se pueden ajustar cuatro recursos para completar el proyecto antes de una fecha límite: tiempo, costo, calidad y alcance. Cuando se incluyen adecuadamente estas cuatro variables de control en la planeación, hay un estado de equilibrio entre los recursos y las actividades necesarias para completar el proyecto. En el capítulo 3 se tratan con mayor detalle estos recursos y cómo se pueden ajustar.

**Cuatro prácticas esenciales de XP** Cuatro prácticas esenciales distinguen notablemente a XP de otros enfoques, y por consiguiente hacen extremo a XP: liberación limitada; semana de trabajo de 40 horas; alojar al cliente en el sitio, y el uso de la programación en parejas.

1. La liberación limitada significa que el equipo de desarrollo reduce el tiempo entre las liberaciones de su producto. En lugar de liberar una versión completamente desarrollada por un año, usando la práctica de liberación limitada reducirán el tiempo de liberación incluyendo primero las características más importantes, liberando ese sistema o producto y mejorándolo después.
2. La semana de trabajo de 40 horas significa que los equipos de desarrollo de XP apoyan conscientemente una práctica esencial cultural en la cual el equipo labora intensamente en conjunto durante una semana típica de 40 horas de trabajo. Como consecuencia a esta práctica, la cultura refuerza la idea de que trabajar horas extras por más de una semana en un turno es muy malo para la salud del proyecto y los diseñadores. Esta práctica esencial intenta motivar a los miembros del equipo a trabajar intensamente durante sus horas de trabajo, y después tomar un descanso para que cuando vuelvan estén relajados y menos estresados. Esto ayuda a los miembros del equipo a identificar los problemas más rápidamente, y previene errores costosos y omisiones debido al desempeño ineficaz o desgastante.
3. El cliente en el sitio significa que un usuario experto en los aspectos de negocios del proyecto en desarrollo está en el sitio durante este proceso. Esta persona es fundamental para el proyecto, escribe las historias del usuario, comunica a los miembros del equipo, ayuda a priorizar y equilibrar las necesidades de largo plazo del negocio y toma decisiones sobre qué características se deben incluir primero.
4. La programación en parejas es una práctica esencial importante. Significa que usted trabaja con otro programador de su propia elección. Ambos codifican, ambos aplican las pruebas. Con frecuencia, la persona con más experiencia tomará el mando de la codificación inicial, pero conforme se involucra la persona con menos experiencia, cualquiera de los dos que tenga la visión más clara de la meta trabajará en la codificación. Cuando le pide a otra persona que trabaje con usted, el protocolo de la programación en parejas dice que está obligada a aceptar. Trabajar con otro programador le ayuda a aclarar su pensamiento. Las parejas cambian con frecuencia, sobre todo durante la fase de exploración del proceso de desarrollo. La programación en parejas ahorra tiempo, reduce las distracciones, activa la creatividad y es una forma divertida de programar.



**FIGURA 6.10**

Las prácticas esenciales de XP se interrelacionan con los recursos, actividades y valores de XP.

La figura 6.10 muestra cómo se interrelacionan y dan soporte las prácticas esenciales de XP con las actividades, los recursos y los valores de XP.

## PROCESO Y HERRAMIENTAS DEL DESARROLLO DE XP

Ahora que ha aprendido algo de las actividades, los recursos y las prácticas esenciales de XP, podemos poner en práctica dicho conocimiento. Esta sección describe el proceso de desarrollo de la programación extrema, explica los detalles involucrados en el registro de las historias del usuario y examina algunas de las herramientas disponibles actualmente para desarrollar sistemas con un enfoque de XP.

**Proceso de desarrollo de XP** En el capítulo 1 aprendió sobre el SDLC y sus diversas fases. La programación extrema también posee un proceso de desarrollo, pero es mucho más interactivo, reiterativo e integrador que el del SDLC. Sin embargo, XP no guía al desarrollador a través

de las fases. Más bien es incremental y con frecuencia las actividades se hacen al mismo tiempo. Observe que diario se hacen muchos pasos del ciclo de XP. Esto contrasta claramente con el SDLC, mismo que procede a un paso mucho más lento y para el cual algunas actividades (el análisis de requerimientos, probar, etc.) deben ser completadas en fases distintas. El proceso de XP permite lograr las actividades.

Las cinco fases del proceso de desarrollo de XP son la exploración, planeación, iteraciones a la primera versión, puesta en producción y mantenimiento (véase el capítulo 3 para una descripción detallada de las fases). La sección siguiente describe específicamente cómo se despliega una sesión típica de trabajo de XP durante el proceso de desarrollo. Por ejemplo, el proceso típico sería asumir una tarea que se relaciona directamente a una característica del sistema que un cliente desea, probarla, implementarla en un diseño existente, e integrarla, todo esto durante un solo episodio del desarrollo. El día podría empezar al analizar el reporte de un usuario pedido en una cédula, en el cual se registra una tarea específica. Durante un informe, pedido para la reunión, usted hace unas preguntas sobre el trabajo hecho el día anterior que podría ayudar en esta tarea. Después le pide a otro programador que le ayude en la tarea.

Se consulta rápidamente a otros expertos en el sitio que podrían conocer las respuestas a las preguntas específicas. Después se consulta el paquete existente de casos de prueba. Probablemente algunos se aplicarán y algunos otros se deberían escribir.

El próximo paso sería anotar la siguiente tarea en una lista especial. Después podría escribir un caso de prueba para cualquier cosa que esté intentando averiguar. Finaliza y lo ejecuta. Probablemente fallará. Usted y su compañero miran otros casos de prueba y depuran lo que escribieron. Continúa con el siguiente caso de prueba, y el siguiente. Finalmente, usted llega al último elemento de su lista de pendientes, que sería reestructurar los otros casos de pruebas; y lo hace.

Usted carga la versión actualizada y los cambios. Después aplica todos los casos de prueba, depura cualquiera que no se ejecute y repara el código. Cuando lo ejecuta nuevamente y funciona, usted ha terminado. Entonces el código se puede liberar.

Aún se podría estar preguntando cómo iniciar una tarea de desarrollo de XP. Un autor experimentado consigue ir directamente al corazón del problema y sólo siendo ligeramente gracioso, escribió: “Escoja su peor problema, resuélvalo con XP. Cuando deje de ser su peor problema, repita” (Wells, citado por Beck, 2000, p. 123). De esta forma, está mostrando una gran valentía. Se está enfocando en resolver primero su problema más urgente, y está aplicando las estrategias de desarrollo de XP para trabajar a través del problema de probar, codificar, escuchar, diseñar e integrar. Está completando todas las tareas del desarrollo de XP en cada asignación de la programación diaria y está reconociendo que el proceso de mejorar el sistema y dirigirse a los problemas graves simple y directamente son la clave para el éxito.

**Cómo escribir las historias de XP** Aún y cuando el título de esta sección sea “Cómo escribir las historias de XP”, el énfasis en la creación de las historias del usuario está en la interacción hablada entre desarrolladores y usuarios, no en la comunicación escrita. En las historias del usuario, el desarrollador ante todo busca identificar los requerimientos valiosos del usuario de negocios. Generalmente los usuarios estarán ocupados diariamente en las conversaciones con los desarrolladores sobre el significado de las historias del usuario que han escrito. Estas conversaciones frecuentes son interacciones determinadas que tienen como su meta la prevención de malos entendidos o malas interpretaciones de los requerimientos del usuario. Por lo tanto, las historias del usuario sirven como recordatorios para los desarrolladores que deben sostener conversaciones seguras para dichos requerimientos.

El siguiente es un ejemplo de una serie de historias escritas para una aplicación de comercio electrónico en línea para un comerciante de libros, CDs y otros productos del medio. Las historias dan un cuadro bastante completo de lo que se necesita en cada una de las fases en el proceso de compra, pero dichas historias son muy cortas y fáciles de comprender. El objetivo

aquí es sacar a relucir todas las necesidades e intereses de la tienda en línea. Aunque no hay suficientes reportes para empezar a programar, un desarrollador de XP podría empezar a ver bastante claro el cuadro global para empezar a estimar lo que se lleva para completar el proyecto. Las historias son como sigue:

**Dé la bienvenida al cliente.**

*Si el cliente ha estado en este sitio antes usando esta misma computadora, déle la bienvenida de nuevo a la tienda en línea.*

**Muestre los aspectos especiales en la página de inicio.**

*Muestre cualquier libro nuevo u otros productos que se han introducido recientemente. Si se identifica al cliente, ajuste las recomendaciones al gusto de ese cliente en particular.*

**Busque el producto deseado.**

*Incluya un mecanismo de búsqueda eficaz que localizará el producto específico y los productos similares.*

**Muestre los títulos correspondientes y la disponibilidad.**

*Despliegue los resultados de la búsqueda en una nueva página Web.*

**Permítale al cliente pedir mayor detalle.**

*Ofrezca al cliente más detalles del producto, tales como páginas de muestra en un libro, más fotografías de un producto o tocar parte de una pista de un CD.*

**Presente las opiniones del producto.**

*Comparta los comentarios que otros clientes tienen sobre el producto.*

**Ponga un producto en un carrito de compras.**

*Hágalo fácil para el cliente. Que sólo haga clic en un botón que ponga el producto en un carrito de las compras deseadas.*

**Registre el historial de la compra en un archivo.**

*Guarde los detalles del cliente y sus compras en una cookie en la computadora del cliente. También guarde la información de la tarjeta de crédito para una verificación rápida.*

**Sugiera otros libros similares.**

*Incluya fotografías de otros libros que tienen temas similares o fueron escritos por los mismos autores.*

**Proceda a la verificación.**

*Confirme la identidad del cliente.*

**Revise las compras.**

*Permítale al cliente revisar las compras.*

**Continúe la compra.**

*Déle la oportunidad al cliente de hacer compras adicionales al mismo tiempo.*

**Aplique métodos abreviados para una verificación más rápida.**

*Si se conoce la identidad del cliente y la dirección de entrega corresponde, se acelera la transacción aceptando la tarjeta de crédito en el archivo y el resto de las preferencias del cliente, tales como el método de envío.*

**Agregue los nombres y las direcciones de envío.**

*Si la compra es un regalo, el cliente puede agregar el nombre y dirección del destinatario.*

<b>Necesidad u oportunidad:</b>		Aplicar métodos abreviados para una rápida verificación.				
<b>Historia:</b>		Si la identidad del cliente es conocida y la dirección de entrega corresponde, acelere la transacción aceptando la tarjeta de crédito en el archivo y el resto de las preferencias del cliente como la forma de envío.				
<b>Actividades:</b>		<b>Muy abajo</b>	<b>Abajo del promedio</b>	<b>Promedio</b>	<b>Arriba del promedio</b>	<b>Muy arriba</b>
	Codificar			✓		
	Probar			✓		
	Escuchar			✓		
<b>Recursos:</b>	Diseñar					
	Tiempo				✓	
	Costo		✓		✓	
	Calidad				✓	
	Alcance			✓		

**FIGURA 6.11**

Las historias de los usuarios se pueden registrar en tarjetas. Dichas historias deben ser bastante breves para que un analista determine qué características de los sistemas son necesarias.

#### Ofrezca opciones de envío.

*Permitale al cliente escoger un método de envío basado en el costo.*

#### Complete la transacción.

*Termine la transacción. Pida la confirmación de tarjeta de crédito si la dirección enviada es diferente de la dirección del cliente en el archivo.*

Como puede ver fácilmente, no hay escasez de historias. El analista de XP necesita escoger unas historias, completar la programación y liberar un producto. Una vez que esto se hace, se seleccionan más historias y se libera una nueva versión hasta que se incluyan todas las historias en el sistema (o el analista y cliente estén de acuerdo que a una historia particular le falta mérito, o no es urgente, y por ello no es necesario incluirla).

En la figura 6.11 se muestra el ejemplo de una historia de usuario como podría aparecerle a un desarrollador de XP. En las cédulas (o electrónicamente), un analista podría identificar primero la necesidad u oportunidad, y seguir con una descripción breve de la historia. El analista podría aprovechar la oportunidad de iniciar pensando ampliamente sobre las actividades que necesitan ser completadas así como también los recursos que tomará para terminar el proyecto. En este ejemplo del comerciante en línea, el analista indica que diseñar la actividad tomará un esfuerzo superior, y que se requieren los recursos de tiempo y calidad superiores al promedio. Observe que el analista no está intentando ser más preciso de lo que actualmente es posible en esta estimación, pero aún es un ejercicio útil.

**Herramientas de desarrollo de XP** Hay varias herramientas preferidas por los desarrolladores de XP. Los creadores originales del enfoque de XP trabajaban en SmallTalk, y con el tiempo trasladaron su marco de pruebas unitarias (SUnit) a Java, el cual actualmente se llama JUnit. Hay muchos recursos en Web que le permiten descargar los marcos de prueba xUnit para cualquier lenguaje de desarrollo de software que esté usando.

Los creadores de XP tuvieron cuidado de no mezclar sus principios de XP con herramientas de desarrollo particulares. Esto significa que XP puede ser tan flexible como sea necesario todo el tiempo, y también puede evolucionar con nuevas herramientas que estén disponibles. Las herramientas vienen y van, pero los principios deben permanecer intactos sin tener en cuenta esa variabilidad.

Muchas de las herramientas usadas en el desarrollo de XP son baratas o completamente gratuitas. Hay un sitio Web excelente, SourceForge.net, en el que se puede encontrar la mayoría de las herramientas de desarrollo de software. Como verá al examinar el sitio, la programación extrema se usa en muchos lenguajes y plataformas de software, pero



los dos líderes indiscutibles en lo que se refiere al uso extendido y popularidad son Java y Microsoft .NET.

Hay muchos tipos diferentes de herramientas disponibles que dan soporte a las actividades que necesitaría realizar al hacer el desarrollo de XP. Se incluyen herramientas que facilitan la colaboración tales como Wiki Wiki, Whiteboard, Project Web, NetMeeting e IBM's Rational ProjectConsole. También hay herramientas tales como IBM's Rational ClearCase, Visual Intercept, Compuware Track Record y Bliplt que dan soporte a la administración de defectos.

Los probadores unitarios automatizados, probadores de aceptación y probadores de GUI incluyen JUnit, ComUnit, VUnit, NUnit, httpUnit y Rational Visual Test Tools. DevPartner Code Review ayuda con el control de calidad. Además hay herramientas que ayudan con la medición del sistema y desempeño de componentes tales como Jmeter, JUnitPerf, PerfMon, TrueTime, RealTime y Microsoft Visual Studio Analyzer. También hay herramientas que ayudan con la administración de la configuración del código fuente, incluyendo CVS, Visual Source Safe y PVCS. Por último, hay una clase de herramientas con la cual probablemente ya está familiarizado, los entornos de desarrollo de IBM VisualAge, Microsoft Visual Studio .NET y JBuilder.

## LECCIONES APRENDIDAS DE XP

Varios proyectos de programación extrema se han descrito en libros, artículos y sitios Web. Muchos de ellos han sido exitosos, algunos han fracasado, pero al estudiarlos podemos aprender mucho de ellos, así como también de los valores, principios y prácticas esenciales de XP. A continuación se presentan las seis lecciones más importantes que obtuvimos al examinar XP. En la figura 6.12 se muestra un diagrama de esas seis lecciones.

La primera lección es que la liberación limitada permite que evolucionen los sistemas. Con frecuencia se hacen actualizaciones al producto, y los cambios se incorporan rápidamente. De esta forma se permite al sistema crecer y extenderse para que el cliente lo encuentre útil. Al utilizar la liberación limitada, el equipo de desarrollo reduce el tiempo entre las versiones de su producto, mejorando el producto posteriormente conforme lo exija la situación dinámica.



**FIGURA 6.12**

Hay seis lecciones vitales que se pueden dibujar desde el enfoque del desarrollo de XP para los sistemas.

La segunda lección es que la programación en parejas incrementa la calidad global. Aunque la programación en parejas es polémica, fomenta claramente otras actividades positivas necesarias en el desarrollo de sistemas tales como una comunicación buena, identificarse con el cliente, enfocarse primero en los aspectos más valiosos del proyecto, probar todo el código tal como se desarrolla e integrar el nuevo código después de que sus fases se prueban con éxito.

La tercera lección es que los clientes en el sitio son benéficos tanto para el negocio como para el equipo de XP. Los clientes sirven como una referencia rápida y un control real, y el enfoque del diseño de sistemas siempre se mantendrá en su presencia: los clientes se parecen más a los desarrolladores y los desarrolladores sienten más empatía con los clientes.

La cuarta lección que tomamos de XP es que la semana de trabajo de 40 horas mejora la eficiencia. Incluso los mejores desarrolladores son susceptibles a cometer errores y se desgastan si trabajan demasiado duro durante un periodo prolongado. Sin embargo, cuando el equipo de desarrollo está junto cada momento cuenta. ¡Trabajar a un ritmo sostenible es mucho más provechoso para la vida del proyecto, la vida del sistema y la vida del desarrollador! Todos conocemos la moraleja de la liebre y la tortuga.

La quinta lección que deducimos de XP es que los recursos y actividades equilibrados dan soporte a los objetivos del proyecto. Administrar un proyecto no sólo significa obtener en conjunto todos los recursos y tareas. También significa que el analista se enfrenta a varios pros y contras. Algunas veces el costo podría estar predeterminado, en otro momento de crisis, el tiempo podría ser el factor más importante. Las variables de control de recursos de tiempo, costo, calidad y alcance necesitan equilibrarse adecuadamente con las actividades de codificar, diseñar, probar y escuchar.

La última lección que tomamos de la programación extrema es que los valores de XP son importantes para su éxito. Es fundamental para el éxito del proyecto que los analistas incluyan incondicionalmente los valores de comunicación, sencillez, retroalimentación y valentía en todo el trabajo que hagan. Este tipo de compromiso personal y en equipo permite al analista tener éxito en situaciones donde, quien posee capacidades técnicas similares pero carece de valores, fallaría. Una verdadera dedicación a estos valores es fundamental para que el desarrollo sea exitoso.

## **MODELADO ÁGIL Y MELÉ (SCRUM)**

El modelado ágil se basa en los valores, al igual que la programación extrema. Además de los valores de comunicación, sencillez, retroalimentación y valentía, se ha agregado un quinto valor: la humildad. Se requiere una corta lección de la historia. En los primeros 25 años de la computación, el software de la computadora fue desarrollado por expertos que con frecuencia creían saber cómo funcionaba un negocio mejor que los expertos del dominio, sus clientes. Con frecuencia fueron llamados los “gurús” de la computadora. Estos gurús eran muy egocentristas e insistieron en que siempre tenían la razón, aun cuando el cliente pensó de forma distinta. Los gurús no tenían la virtud de ser humildes.

Sin embargo, el valor de la humildad es crítico. Los modeladores ágiles son analistas de sistemas que hacen sugerencias, expresan opiniones, pero no insisten en que siempre tienen la razón. Están seguros de sí mismos para permitirles a sus clientes cuestionar, criticar y algunas veces quejarse del sistema que están desarrollando. Reconocen que pueden aprender de sus clientes. Después de todo, los clientes son los que han tenido la experiencia de operar el negocio durante un periodo largo. También los clientes son responsables del resultado final. Existen mientras la compañía tiene buenas épocas. Los consultores van y vienen, pueden cambiar mucho.

El modelado ágil también abarca un conjunto de principios esenciales. Además de los principios esenciales de la programación extrema, el modelado ágil agrega principios tales como “modelar con un propósito”, “el software es su meta principal” y “viajar con poco equipaje”, una forma de decir que poca documentación es suficiente.

El modelado es una palabra clave en los métodos ágiles. A diferencia de XP, el modelado ágil aprovecha la oportunidad de crear modelos. Éstos pueden ser modelos lógicos tales como diseños de sistemas, o modelos como prototipos descritos anteriormente en este capítulo. Un proceso típico de modelado ágil podría ser algo como lo siguiente:

1. Escuche las historias de usuario del cliente (parecidas a las historias de usuario de XP).
2. Dibuje un modelo de flujo de trabajo lógico para tener una perspectiva de las decisiones del negocio representadas en la historia del usuario.
3. Cree nuevas historias del usuario basadas en el modelo lógico.
4. Desarrolle algunos prototipos de muestra. Con ello, muestre al cliente qué clase de interfaz tendrá.
5. Usando la retroalimentación de los prototipos y los diagramas de flujo de trabajo lógicos, desarrolle el sistema hasta que cree un modelo de datos físico.

Ágil es la otra palabra clave en el modelado ágil. Ágil significa maniobrabilidad. Los sistemas actuales, sobre todo aquellos que se basan en Web, representan una doble demanda: liberar el software tan pronto como sea posible y mejorarlo continuamente para agregar nuevas características. El analista de sistemas debe tener la habilidad y métodos para crear las aplicaciones dinámicas, contextuales, escalables y evolutivas. El modelado ágil tal como un método de aceptación de cambios, no es diferente a la programación extrema.

Un enfoque ágil se conoce como “*melé*”. La palabra *melé* se toma de la posición de arranque en rugby en la cual los equipos se ponen frente a frente y pelean por la posesión del balón. En realidad *melé* se refiere al trabajo en equipo, similar a lo que se necesita hacer en un juego de rugby.

Tal como los equipos de rugby empezarán un juego con una estrategia global, los equipos de desarrollo empiezan el proyecto con un plan de alto nivel que puede cambiarse sobre la marcha conforme avance el “juego”. Los miembros del equipo de desarrollo de sistemas comprenden que el éxito del proyecto es más importante, y su éxito individual es secundario. El líder del proyecto tiene alguna, pero no mucha, influencia en los detalles. Más bien, las tácticas del juego se dejan a los miembros del equipo, como si estuvieran en el campo. El equipo de sistemas trabaja dentro de un horario estricto (30 días para el desarrollo), así como un equipo de rugby jugaría limitándose estrictamente al tiempo de un juego.

Podemos describir los componentes de la metodología de *melé* como:

1. Productos atrasados, en donde se deriva una lista de las especificaciones del producto.
2. Atrasos para arranques, una lista cambiante dinámicamente de tareas a ser completadas en el próximo arrancón.
3. Arrancón, un periodo de 30 días en el cual el equipo de desarrollo transforma el atraso, en software que puede demostrarse.
4. *Melé* diaria, una reunión breve donde la comunicación es la regla número uno. Los miembros del equipo necesitan explicar lo que hicieron desde la última reunión, si encontraron obstáculos y qué planean hacer antes de la próxima *melé* diaria.
5. Software funcional de demostración que puede demostrarse al cliente.

De hecho, *melé* es una metodología de alta intensidad. Es sólo uno de los enfoques que adoptan la filosofía del modelado ágil.

---

## RESUMEN

La elaboración de prototipos es una técnica útil de recopilación de información para complementar el ciclo de vida del desarrollo tradicional de sistemas. Cuando los analistas de sistemas usan la elaboración de prototipos, están buscando las reacciones del usuario, sugerencias, innovaciones y la revisión planeada para mejorar el prototipo, y por consiguiente modificar los planes del sistema con un gasto e interrupción mínimos. Los sistemas que

apoyan la toma de decisiones semiestructurada (como lo hacen los sistemas de apoyo a la decisión) son los primeros candidatos para la elaboración de prototipos.

El término *elaboración de prototipos* acepta varios significados diferentes, de los cuales cuatro se usan comúnmente. La primera definición de la elaboración de prototipos es la de construir un prototipo como un sistema corregido. Una segunda definición es la de un prototipo no funcional que se usa para probar ciertos aspectos del diseño. Como tercera definición está la de crear el primer prototipo de una serie que es totalmente funcional. Esta clase de prototipo es útil cuando se planean muchas instalaciones del mismo sistema de información (bajo condiciones similares). La cuarta clase de la elaboración de prototipos es un prototipo con características seleccionadas que tiene algunas, pero no todas, las características principales del sistema. Usa módulos independientes como los blocks para construcción de manera que si las características del prototipo elaborado son exitosas, se puedan mantener e incorporarse en el sistema final.

Los cuatro lineamientos principales para desarrollar un prototipo son: (1) trabajar en módulos manejables; (2) construir rápidamente el prototipo; (3) modificar el prototipo, y (4) poner énfasis en la interfaz de usuario.

Una desventaja de los prototipos es que administrar el proceso de la elaboración de prototipos es difícil debido a la rapidez del proceso y a sus muchas iteraciones. Una segunda desventaja es que un prototipo incompleto podría ser forzado a colocarse en servicio como si fuera un sistema completo.

Aunque la elaboración de prototipos no siempre es necesaria o deseable, se debe observar que hay tres ventajas principales relacionadas con su uso: (1) la aptitud para cambiar a tiempo el sistema en su desarrollo, (2) la oportunidad de detener el desarrollo de un sistema que no está funcionando y (3) la posibilidad de desarrollar un sistema que se ajuste más estrechamente a las necesidades y expectativas de los usuarios.

Los usuarios tienen que desempeñar un papel diferente en el proceso de la elaboración de prototipos. Su ocupación principal debe ser interactuar con el prototipo a través de la experimentación. Los analistas de sistemas deben trabajar sistemáticamente para identificar y evaluar las reacciones de los usuarios al prototipo.

Un uso particular de la elaboración de prototipos es el de desarrollo rápido de aplicaciones, o RAD. Éste es un enfoque orientado a objetos con tres fases: planeación de requerimientos, taller de diseño del RAD e implementación.

La programación extrema (XP) es un enfoque de desarrollo de software que toma lo que generalmente designamos como buenas prácticas de desarrollo de software y las lleva al extremo. XP define con rapidez un plan global, desarrolla, libera rápidamente el software y después lo revisa de manera continua para agregarle características adicionales. Los programadores de XP trabajan en parejas para desarrollar sistemas de calidad.

Los cuatro valores de XP que son compartidos por el cliente comercial así como también por el equipo de desarrollo son comunicación, sencillez, retroalimentación y valentía. Los cinco principios básicos de XP consisten en proporcionar una retroalimentación rápida; adoptar la simpleza al abordar una nueva tarea de programación; cambiar el código, el diseño, e incluso al equipo de desarrollo, de manera gradual; aceptar el cambio como un estado normal del trabajo, y hacer un trabajo de calidad. Las actividades de XP incluyen codificar, probar, escuchar y diseñar. Los recursos disponibles incluyen tiempo, costo, calidad y alcance.

Las cuatro prácticas esenciales de XP son: (1) liberación limitada; (2) semana de trabajo de 40 horas; (3) cliente en el sitio, y (4) programación en parejas. Dichas prácticas distinguen a la programación extrema de otros procesos de desarrollo de sistemas. Las cinco fases amplias en el proceso de desarrollo de XP son la exploración, planeación, las iteraciones a la primera versión, puesta en producción y mantenimiento.

El proceso de desarrollo de XP incluye seleccionar una tarea que se relaciona directamente a una característica deseada por el cliente que se basa en las historias del usuario; escoger a un compañero de programación; seleccionar y escribir los casos de prueba adecuados; escribir el código; aplicar los casos de prueba; depurar el código hasta que se apliquen



“Gracias a Dios ésta es la época del año en que todo es nuevo. Yo amo la primavera; es la época más excitante aquí en MRE. Los árboles son tan verdes, con hojas de diversas tonalidades. Tantos nuevos proyectos por hacer, también; tantos nuevos clientes por conocer. Es realmente excitante. Me recuerda la elaboración de prototipos. O lo que sé sobre la elaboración de prototipos. Es algo nuevo y fresco, una manera rápida de averiguar lo que está pasando.”

“De hecho, creo que ya tenemos algunos prototipos por aquí. Lo mejor de todo es que pueden cambiar. No conozco a nadie que realmente haya quedado satisfecho con un primer prototipo. Pero es divertido involucrarse con algo que se hace rápidamente, y que cambiará.”

## PREGUNTAS DE HYPERCASE

1. Localice algún prototipo propuesto que se use actualmente en uno de los departamentos de MRE. Sugiera algunas modificaciones que harían este prototipo aún más sensible a las necesidades de la unidad.
2. Usando un procesador de texto, construya un prototipo no funcional para un Sistema de Informes sobre Proyectos para la Unidad de Capacitación. Si dispone de un programa de hipertexto, intente incorporar alguna funcionalidad agregando menús con opciones que se puedan seleccionar. *Sugerencia:* tome como base para su diseño las pantallas de ejemplo de los capítulos 11 y 12.

**Sistema global de ingeniería de administración**

Recursos de edición

Número del recurso

Nombre del recurso

Ubicación del recurso

Supervisor del recurso

Número telefónico del recurso

Disponibilidad del recurso

Honorarios del recurso

Base para el pago de honorarios al recurso

Número de pedido

**FIGURA 6.HC1**

Una de las muchas pantallas de prototipo que se encuentran en HyperCase.

todos los casos de prueba; implementarlo con el diseño existente, e integrarlo a lo que actualmente existe.

Hay seis lecciones que aprender del enfoque de XP. La primera lección es que las liberaciones en corto permiten evolucionar a los sistemas. La segunda lección es que la programación en parejas refuerza la calidad global. La tercera lección es que los clientes en el sitio y el equipo de XP se benefician mutuamente. La cuarta lección es que la semana de trabajo de 40 horas mejora la eficiencia. La quinta lección es que los recursos y actividades equilibrados apoyan las metas del proyecto. La última lección que tomamos de la programación extrema es que los valores de XP son fundamentales para el éxito.

El modelado ágil abarca un conjunto de principios básicos. Un valor que los modeladores ágiles poseen es la humildad. Además de los principios esenciales de la programación extrema, el modelado ágil agrega principios tales como “modelar con un propósito”, “el software es su meta principal” y “viajar con poco equipaje”, una forma de decir que poca documentación es suficiente. Una forma de implementar el modelado ágil es con la metodología de *melé*.

## PALABRAS Y FRASES CLAVE

aceptar el cambio	metodología <i>melé</i>
adoptar la sencillez	modelado ágil
cambio incremental	modificar el prototipo
cliente en el sitio	primer prototipo de una serie
desarrollo rápido de aplicaciones (RAD)	principios de XP
énfasis en la interfaz de usuario	programación en parejas
fase de exploración	programación extrema (XP)
fase de mantenimiento	prototipo
fase de planeación	prototipo corregido
fase de planeación de requerimientos	prototipo de características
fase de puesta en producción	seleccionadas
historias del usuario	prototipo no funcional
implementación	retroalimentación rápida
intervención del usuario en la elaboración de prototipos	semana de trabajo de 40 horas
iteraciones a la primera fase de liberación	taller de diseño del RAD
liberación limitada	trabajar en módulos manejables
	valores de XP

## PREGUNTAS DE REPASO

1. ¿Cuáles son los cuatro tipos de información que busca el analista en la elaboración de prototipos?
2. ¿Qué significa el término *prototipo corregido*?
3. Defina un prototipo que es un modelo a escala no funcional.
4. Proporcione un ejemplo de un prototipo que es un primer modelo a escala completa.
5. Defina lo que significa un prototipo que es un modelo con algunas, pero no todas, las características principales.
6. Haga una lista de las ventajas y desventajas de usar la elaboración de prototipos para *reemplazar* el ciclo de vida del desarrollo tradicional de sistemas.
7. Describa cómo se puede usar la elaboración de prototipos para aumentar el ciclo de vida del desarrollo tradicional de sistemas.
8. ¿Cuáles son los criterios para decidir si se debe hacer un prototipo de un sistema?
9. Mencione cuatro lineamientos que el analista debe observar en el desarrollo de un prototipo.
10. ¿Cuáles son los dos problemas principales identificados en la elaboración de prototipos?
11. Mencione las tres ventajas principales de utilizar la elaboración de prototipos.
12. ¿Cómo puede un prototipo de un sitio Web interactivo facilitar el proceso de la elaboración de prototipos? Conteste en un párrafo.

13. ¿Cuáles son las tres formas en que un usuario puede ser de ayuda en el proceso de la elaboración de prototipos?
14. Defina lo que significa RAD.
15. ¿Cuáles son las tres fases del RAD?
16. Defina la programación extrema.
17. ¿Cuáles son los cuatro valores que deben compartir el equipo de desarrollo y los clientes de negocios cuando se toma un enfoque de programación extrema?
18. ¿Cuáles son los cinco principios básicos de la programación extrema?
19. ¿Cuáles son las cuatro prácticas principales del enfoque de desarrollo de XP?
20. Delinee los pasos típicos en un episodio de desarrollo de XP.
21. ¿Qué es una historia de usuario? ¿Es principalmente escrita o hablada? Elija su opción, luego apoye su respuesta con un ejemplo.
22. Mencione las herramientas de software que pueden ayudar al desarrollador a hacer una variedad de pruebas de código.
23. ¿Cuáles son las seis lecciones tomadas de la experiencia con los esfuerzos del desarrollo de XP?
24. Compare y contraste el modelado ágil con el enfoque de XP.
25. ¿Qué es *melé*?

## PROBLEMAS

1. Como parte de un proyecto de sistemas extenso, Clone Bank de Clone, Colorado, necesita su ayuda para crear un nuevo formulario de informe mensual para las cuentas de cheques y ahorros de sus clientes. El presidente y el vicepresidente están en sintonía con lo que dicen los clientes en la comunidad. Piensan que sus clientes quieren un resumen de la cuenta de cheques que se parezca al que ofrecen los otros tres bancos de la ciudad. Sin embargo, no están dispuestos a hacer ese formulario sin un resumen formal de retroalimentación del cliente que apoye sus decisiones. La retroalimentación no se usará para cambiar el formulario del prototipo de ninguna forma. Ellos quieren que usted envíe el prototipo de un formulario a un grupo y el formulario viejo a otro grupo.
  - a. En un párrafo explique por qué probablemente no es importante crear un prototipo de sistema para el nuevo formulario bajo estas circunstancias.
  - b. En un segundo párrafo explique bajo qué situación sería aconsejable crear un prototipo para el nuevo formulario.
2. Por muchos años C. N. Itall ha sido analista de sistemas para la Tun-L-Vision Corporation. Cuando usted se integró al equipo de análisis de sistemas y sugirió la elaboración de prototipos como parte del SDLC para un proyecto actual, C. N. dijo: “Seguro, pero no puede prestar atención a lo que dicen los usuarios. No tienen idea de lo que quieren. Elaboraré el prototipo, pero no ‘observaré’ a ningún usuario”.
  - a. Tan cuidadosamente como sea posible, de manera que no moleste a C. N. Itall, haga una lista de las razones que apoyan la importancia de observar las reacciones, sugerencias e innovaciones del usuario en el proceso de la elaboración de prototipos.
  - b. En un párrafo describa lo que podría pasar con los sistemas siguientes si se desarrollara parte de un sistema y no se incorporara ninguna retroalimentación del usuario.
3. “Cada vez que pienso que he capturado los requerimientos de información del usuario, éstos ya han cambiado. Es lo mismo que intentar pegarle a un blanco en movimiento. La mitad del tiempo, creo que ellos mismos aún no saben lo que quieren”, exclama Flo Chart, analista de sistemas para 2 Good 2 Be True, una empresa que inspecciona el producto utilizado para las divisiones de marketing de varias compañías industriales.
  - a. En un párrafo, explique a Flo Chart cómo la puede ayudar la elaboración de prototipos a definir bien los requerimientos de información de los usuarios.
  - b. En un párrafo, comente sobre la observación de Flo: “La mitad del tiempo, creo que ellos mismos aún no saben lo que quieren”. Asegúrese de explicar cómo puede ayudar realmente la elaboración de prototipos a los usuarios a entender y articular mejor sus propios requerimientos de información.

- c. En un párrafo, sugiera cómo un sitio Web interactivo que presente un prototipo podría resolver las preocupaciones de Flo sobre la captura de los requerimientos de información del usuario.
4. Harold, un gerente de departamento de la cadena de tiendas Sprocket's Gifts, piensa que la construcción de un prototipo puede significar sólo una cosa: un modelo a escala no funcional. También cree que esta forma es demasiado incómoda para hacer prototipos de los sistemas de información y por ello es renuente a hacerlo.
  - a. Brevemente (en dos o tres párrafos) compare y contraste los otros tres tipos de elaboración de prototipos que son posibles para que Harold comprenda lo que puede significar la elaboración de prototipos.
  - b. Harold tiene la opción de implementar un sistema, probarlo y después instalarlo en otras cinco ubicaciones de Sprocket, si tiene éxito. Nombre un tipo de elaboración de prototipos que encajaría bien con este enfoque, y en un párrafo respalde su opción.
5. "¡He tenido la idea del siglo!", clama Bea Kwicke, nuevo analista de sistemas de su grupo de sistemas. "Saltémonos toda esta basura del SDLC y tan sólo hagamos un prototipo de todo. Nuestros proyectos irán mucho más rápido, nos ahorraremos tiempo y dinero, y todos los usuarios sentirán como si les pusiéramos atención en lugar de alejarnos sucesivamente durante meses y no hablar con ellos."
  - a. Liste las razones que usted (como miembro del mismo equipo que Bea) le daría para disuadirla de intentar desechar el SDLC y hacer un prototipo de cada proyecto.
  - b. Bea no está muy de acuerdo con lo que ha dicho. Para animarla, use un párrafo para explicar las situaciones que piensa se presentarían en la elaboración de prototipos.
6. El comentario siguiente se oyó por casualidad en una reunión entre los gerentes y un equipo de análisis de sistemas en la compañía Fence-Me-In: "Usted nos dijo que el prototipo estaría listo hace tres semanas. ¡Aún lo estamos esperando!"
  - a. En un párrafo, comente la importancia de entregar rápidamente una parte de un prototipo elaborado de un sistema de información.
  - b. Mencione tres elementos del proceso de la elaboración de prototipos que se deben controlar para asegurar la entrega puntual del prototipo.
  - c. ¿Cuáles son algunos de los elementos del proceso de la elaboración de prototipos que son difíciles de manejar? Lístelos.
7. Examine la recopilación de historias de usuario del comerciante en línea mostrado anteriormente en el capítulo. A la tienda de medios en línea ahora le gustaría que usted agregue algunas características a su sitio Web. Siguiendo el formato mostrado en la figura 6.11 escriba una historia de usuario para las características listadas debajo:
  - a. Incluya anuncios desplegables.
  - b. Ofrezca compartir los detalles de las compras del cliente con sus amigos.
  - c. Extienda la oferta para permitir comprar otros artículos.
8. Visite el sitio Web de las herramientas Palm en [www.palmgear.com](http://www.palmgear.com). Explore el sitio Web y haga un reporte de una docena de historias de usuario breves para mejorar el sitio Web.
9. Visite el sitio Web de techtv en [www.techtv.com](http://www.techtv.com) y haga un reporte de una docena de historias de usuario breves para mejorar el sitio Web.
10. Usando las historias que escribió en el problema 7, pase por las cinco fases del proceso de desarrollo de XP y describa lo que sucede en cada una de ellas.

## PROYECTOS DE GRUPO

1. Divida su grupo en dos subgrupos más pequeños. Haga que el grupo 1 siga los procesos especificados en este capítulo para crear prototipos. Usando una herramienta CASE o un procesador de texto, el grupo 1 debe diseñar dos pantallas de prototipo no funcionales usando la información recopilada en las entrevistas con empleados de Maverick Transport completados en el ejercicio de grupo del capítulo 4. Haga las suposiciones necesarias para crear dos pantallas para despachadores de camiones. El grupo 2 (repre-



sentando los papeles de despachadores) deben reaccionar a las pantallas de prototipo y proporcionar retroalimentación sobre las adiciones y eliminaciones deseadas.

2. Los miembros del grupo 1 deben revisar las pantallas de prototipo basados en los comentarios del usuario que hayan recibido. Los del grupo 2 deben responder con comentarios acerca de qué tan bien se resolvieron sus preocupaciones iniciales en los prototipos refinados.
3. En grupo, expliquen en un párrafo sus experiencias con la elaboración de prototipos para determinar los requerimientos de información.

---

## BIBLIOGRAFÍA SELECCIONADA

- Alavi, M., "An Assessment of the Prototyping Approach to Information Systems Development", *Communications of the ACM*, vol. 27, núm. 6, junio de 1984, pp. 556-563.
- Avison, D. y D. N. Wilson, "Controls for Effective Prototyping", *Journal of Management Systems*, vol. 3, núm. 1, 1991.
- Baird, S., *SAMS Teach Yourself Extreme Programming in 24 Hours*, Indianápolis, IN: SAMS Publishing, 2003.
- Beck, K., *Extreme Programming EXPlained: Embrace Change*, Boston: Addison-Wesley Publishing Co., 2000.
- Beck, K. y M. Fowler, *Planning Extreme Programming*, Boston: Addison-Wesley Publishing Co., 2001.
- Billings, C., M. Billings y J. Tower. *Rapid, Application Development with Oracle Designer/2000*, Reading, MA: Addison-Wesley, 1996.
- Cockburn, A., *Agile Software Development*, Boston: Addison-Wesley Publishing Co., 2002.
- Davis, G. B. y M. H. Olson, *Management Information Systems: Conceptual Foundations, Structure, and Development*, 2a. ed., Nueva York: McGraw-Hill, 1985.
- Dearnley, P. y P. Mayhew, "In Favour of System Prototypes and Their Integration into the Systems Development Cycle", *Computer Journal*, vol. 26, febrero de 1983, pp. 36-42.
- Ghione, J., "A Web Developer's Guide to Rapid Application Development Tools and Techniques", *Netscape World*, junio de 1997.
- Gremillion, L. L. y P. Pyburn, "Breaking the Systems Development Bottleneck", *Harvard Business Review*, marzo-abril de 1983, pp. 130-137.
- Harrison, T. S., "Techniques and Issues in Rapid Prototyping", *Journal of Systems Management*, vol. 36, núm. 6, junio de 1985, pp. 8-13.
- Liang, D., *Rapid Java Application Development Using JBuilder 3*, Upper Saddle River, NJ: Prentice Hall, 2000.
- McBreen, P., *Questioning Extreme Programming*, Boston: Addison-Wesley Publishing Co., 2003.
- McMahon, D., *Rapid Application Development with Visual Basic 6 (Enterprise Computing)*, Nueva York: McGraw-Hill Professional Publishing, 1999.
- , *Rapid Application Development with Visual C++*, Nueva York: McGraw-Hill Professional Publishing, 1999.
- Naumann, J. D. y A. M. Jenkins, "Prototyping: The New Paradigm for Systems Development", *MIS Quarterly*, septiembre de 1982, pp. 29-44.

## 6



ALLEN SCHMIDT, JULIE E. KENDALL Y KENNETH E. KENDALL

**ES HORA DE REACCIONAR**

“Necesitamos saber lo que quieren los usuarios respecto a la salida”, comenta Anna. “Esto nos ayudará a confirmar algunas de nuestras ideas sobre la información que necesitan.”

“Estoy de acuerdo”, contesta Chip. “También nos ayudará a determinar la entrada necesaria. A partir de ahí podemos diseñar las pantallas de entrada de datos correspondientes. Generemos prototipos de los informes y las pantallas y obtengamos retroalimentación de los usuarios. ¿Por qué no usamos Microsoft Access para crear rápidamente pantallas e informes? Conozco bien el software.”

Anna empieza por desarrollar el prototipo del informe PREVENTIVE MAINTENANCE REPORT. Basada en los resultados de las entrevistas, comienza a generar el informe que cree que necesitará Mike Crowe.

“Este informe se debe usar para predecir cuándo se les debe dar mantenimiento preventivo a las máquinas”, piensa Anna. “Me parece que Mike necesita saber *qué* máquina tiene que realizar el trabajo, así como también *cuándo* se debe programar el trabajo. Ahora veamos, ¿qué información identificaría claramente la máquina? El número de inventario, la marca y el modelo identificarían la máquina. Pienso que la sala y el campus se deben incluir para localizar rápidamente la máquina. Una fecha de mantenimiento calculada le indicaría a Mike cuándo se debe completar el trabajo. ¿En qué secuencia debe estar el informe? Tal vez la más útil sería por ubicación.”

En la figura E6.1 se muestra el prototipo del informe PREVENTIVE MAINTENANCE REPORT que presenta el informe terminado. Observe que XXXXXXXX y las fechas genéricas se usan para indicar en dónde se deben imprimir los datos. Se incluyen las ubicaciones reales del campus y la sala así como también los números de inventario. Éstos son necesarios para que Microsoft Access realice la impresión del grupo.

El prototipo del informe se termina rápido. Después de imprimir la última copia, Anna lleva el informe a Mike Crowe y Dot Matrix. Mike Crowe está entusiasmado con el proyecto y quiere saber cuándo estará en producción el informe. Dot también está impresionada.

Surgen varios cambios. Mike quiere un área para escribir la fecha límite del mantenimiento preventivo de manera que el informe se pueda usar para reingresar las fechas en la computadora. Dot quiere que el número de informe asignado por el control de datos aparezca en la parte superior del formulario con el fin de que sirva de referencia. También sugiere que el título del informe se cambie a WEEKLY PREVENTIVE MAINTENANCE REPORT. Los próximos pasos son modificar el prototipo del informe para reflejar los cambios recomendados y después dárselo a Mike y Dot para que revisen el resultado.

El informe se modifica e imprime fácilmente. Dot está contenta con el resultado final. “En realidad éste es un buen método para diseñar el sistema”, comenta. “Es muy agradable sentir que somos parte del proceso de desarrollo y que nuestras opiniones cuentan. Estoy empezando a sentir bastante confianza de que el sistema final será lo que siempre hemos querido.”

Mike expresa un cumplido similar: “Esto hará mucho más fluido nuestro trabajo. Elimina la necesidad de adivinar qué máquinas requieren mantenimiento. Y ordenarlas por sala es una buena idea. No tendremos que invertir tanto tiempo en regresar a las salas para trabajar en las máquinas”.

Chip toma nota de cada una de estas modificaciones en un Formulario de Evaluación de Prototipo. Este formulario ayuda a Chip a organizarse y documentar el proceso de la elaboración de prototipos. (En la figura 6.3 se muestra un ejemplo de este formulario.)

**Preventive Maintenance Report**  
Week of 1/11/2004

1/11/04 Page 1 of 1

Campus Location	Room Location	Inventory Number	Brand Name	Model	Last Preventive Maintenance Date	Done
Central Administration	11111	84004782	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	11/4/03	___
Central Administration	11111	90875039	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	10/24/03	___
Central Administration	11111	93955411	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	11/4/03	___
Central Administration	11111	99381373	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	10/24/03	___
Central Administration	22222	10220129	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	10/24/03	___
Central Administration	99999	22838234	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	10/24/03	___
Central Administration	99999	24720952	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	10/24/03	___
Central Administration	99999	33453403	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	11/4/03	___
Central Administration	99999	34044449	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	11/4/03	___
Central Administration	99999	40030303	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	11/4/03	___
Central Administration	99999	47403948	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	10/24/03	___
Central Administration	99999	56620548	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	11/4/03	___
Central Computer Science	22222	34589349	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	10/24/03	___
Central Computer Science	22222	38376910	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	10/24/03	___
Central Computer Science	22222	94842282	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	10/24/03	___
Central Computer Science	99999	339393	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	11/4/03	___
Central Zoology	22222	11398423	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	10/24/03	___
Central Zoology	22222	28387465	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	11/4/03	___
Central Zoology	99999	70722533	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	10/24/03	___
Central Zoology	99999	99481102	Xxxxxxxxxxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	10/24/03	___

**FIGURA E6.1**

Prototipo del informe PREVENTIVE MAINTENANCE REPORT. Este informe debe modificarse.

A continuación, Chip y Anna centran su atención en crear los prototipos de pantalla. “Como a mí me gusta el aspecto del hardware del sistema, ¿por qué no empiezo a trabajar en el diseño de la pantalla ADD NEW COMPUTER?”, pregunta Chip.

“Me parece bien”, contesta Anna. “Me enfocaré en los aspectos del software.”

Chip analiza con Dot y Mike los resultados de las entrevistas detalladas. Recopila una lista de los elementos que cada usuario necesitaría al agregar una computadora. Otros elementos, tales como la información de ubicación y mantenimiento, actualizarían posteriormente el archivo COMPUTER MASTER, después de instalar la máquina.

En la figura E6.2 se muestra el prototipo de la pantalla ADD NEW COMPUTER que se creó con la característica de formularios de Access. En la parte superior de la pantalla están la fecha y la hora actuales así como el título de la pantalla centrado. Los títulos de los campos se colocan alineados a la izquierda en la pantalla. Se incluyen casillas de verificación para varios campos, así como listas desplegables para el tipo de monitor, impresora y cone-

## 6

Central Pacific University - CPU - [Add New Computers]

2/20/04 Add New Computer 1:01

Inventory Number: 2345 Serial Number: 0000000000000000

Brand Name: 000000000000 Model: 000000000000

Date Purchased: 1/5/04 Purchase Cost: \$3,999.00

Memory Size (MB): 512 Replacement Cost: \$3,999.00

Hard drive (GB): 60 Second Hard Drive: 0

CD ROM Drive: RW Zip Drive Connected: ☒

Monitor: Flat Screen Monitor Network: ☒

Printer: Stylus Multifunction Network Connection: T1 Line

Warranty: ☒

Board Code
Graphics Accelerator
Network Card
Sound Card

Code for computer internal boards

**FIGURA E6.2**

Prototipo de la pantalla ADD NEW COMPUTER. Microsoft Access se utilizó como herramienta para elaborar los prototipos. Las mejoras se pueden hacer en esta etapa.

xiones de red. Se incluye una pequeña tabla de códigos de tarjeta (Board Code) para agregar varias tarjetas internas a una computadora. Se incluyen los botones “Add Record” y “Print”.

“El hecho de tener definidas las tablas de la base de datos seguramente ayuda a hacer rápido los prototipos”, comenta Chip. “No tomó mucho tiempo completar la pantalla. ¿Les gustaría verme probar el prototipo?”

“Claro”, contesta Anna. “Ésta es mi parte favorita de la elaboración de prototipos.”

Chip ejecuta el diseño de la pantalla mientras Anna, Mike y Dot observan. Las listas desplegables y las casillas de verificación le facilitan ingresar los datos exactos.

“Realmente me gusta esto”, dice Dot. “¿Me permites introducir algunos datos?”

“Por supuesto”, contesta Chip. “Trata de agregar datos inválidos y válidos. Y observa los mensajes de ayuda que aparecen en la parte inferior de la pantalla para indicar lo que se debe introducir.”

Anna regresa a su escritorio y hace el diseño de la pantalla ADD SOFTWARE RECORD.

Cuando Anna completa el diseño de la pantalla, le pide a Cher que pruebe el prototipo. Cher teclea información, verifica los valores de las listas desplegables y visualiza los mensajes de ayuda.

“Realmente me gusta el diseño de esta pantalla y su apariencia”, comenta Cher. “Sin embargo, le faltan algunos campos que normalmente se incluyen cuando se introduce un paquete de software, como la marca y modelo de la computadora en que se ejecuta el software, la memoria, monitor y la impresora o plotter requeridos. También quisiera botones para guardar el registro y salir de la pantalla.”

## 6

“No hay problema. Haré los cambios y regresaré contigo”, contesta Anna, tomando algunas notas para sí misma.

Poco tiempo después, Cher prueba nuevamente la pantalla ADD SOFTWARE RECORD. Ésta incluye todas las características que ella requiere. El diseño de la pantalla completa se puede visualizar con Microsoft Access. Observe que una línea separa la información del software y las entradas del hardware.

“Chip, estuve hablando con Dot y mencionó que tiene fondos para poner algo de información en la Web, como parte de un sitio Web unificado que brinde apoyo tecnológico en la CPU”, comenta Anna, apartando la vista de su computadora. “He estado ocupada creando un prototipo para los menús de la página Web y la primera pantalla, donde se puedan reportar los problemas de tecnología. Puesto que Mike es el encargado de resolver problemas, los he invitado a él y a Dot para que revisen el prototipo. ¿Quieres unirte a la sesión?”

“Claro”, contesta Chip. “Estoy interesado en trabajar en el diseño de algunas de las páginas Web.”

Poco tiempo después Mike, Dot y Chip se reúnen con Anna para que les muestre la página Web, ilustrada en la figura E6.3.

“Realmente me gusta el estilo del menú”, comenta Dot. “Las fichas de las características principales en la parte superior son fáciles de usar y me gusta la forma en que cambian de color cuando se hace clic en ellas.”

“Sí, y tener submenús debajo del principal para las características de cada ficha facilita encontrar lo que se está buscando”, agrega Mike. “Sin embargo, tengo algunas sugerencias para reportar los problemas en la página Web. Sería más útil si el área para seleccionar la categoría del problema se pasara a la parte superior de la página. Cada tipo de problema se asigna a un técnico diferente, que cuente con alguna experiencia en esa área. Necesitamos

The screenshot shows a web browser window titled "Central Pacific University: Problem Reporting - Microsoft Internet Explorer". The address bar shows "http://www.cpu.edu/support/problem/problem.html". The page has a navigation menu with links: "Email Setup", "Web Pages", "Support", "Search", "Site Map", and "FAQ". Below this is a sub-menu with "Install Software", "Laptop Checkout", "Report Problem", "Supplies", and "Move Computer". The main heading is "Central Pacific University - Problem Reporting System". The form instructions say: "Please complete the form below and click the **Send Problem** button to notify us of a problem." The form fields include: "First Name" and "Last Name" (text boxes), "Email Address" (text box), "Campus Location" (dropdown menu showing "Central Administration"), "Room Number" (text box), and "Tag Number" (text box with a "Tag Number Help" button). Below these is a "Problem category" section with radio buttons for "Computer", "Software", "Peripheral", "Printer", "Network", and "Other". At the bottom is a large "Problem Description" text area. Two buttons, "Send Problem" and "Clear Form", are at the bottom right. The footer text reads "Email Technical Support System" and "Modified on 02/09/2004 by Mike Crowe".

**FIGURA E6.3**

Prototipo de la página Web PROBLEM REPORTING SYSTEM. Esta página Web necesita algunas mejoras.

## 6

una casilla de verificación adicional para identificar si el equipo o el software con el que estamos trabajando es compatible con Macintosh o con IBM. La ayuda del número de parte (Tag Number) es una gran idea. Muchas personas no saben que cada parte de equipo tiene una pequeña placa de metal con un número único de inventario. Hmmm... Esa gran área azul parece destacar demasiado. Después de todo, sólo es ayuda. Pienso que sería mejor reemplazarla con una pequeña imagen.”

“Pienso que será fácil hacer estos cambios”, comenta Anna.

“Excelente”, contesta Mike. “También sería útil incluir en la página Web el número telefónico de la línea directa del soporte técnico. Si es una verdadera emergencia, podría ayudar a acelerar la resolución del problema. También deberíamos agregar un campo para que introduzcan su número telefónico. Por supuesto, podríamos buscarlo, pero la persona podría estar reportando el problema desde un laboratorio de cómputo u otro lugar fuera de su oficina.”

“¡Buena idea!”, exclama Dot. “Esto será sumamente útil para los profesores y el personal. Creo que deberíamos crear prototipos de todas las páginas Web del sitio. Estoy consciente de que las páginas Web cambian de vez en cuando, ¡pero hagámoslas tan buenas como sea posible desde el principio!”

Anna le echa una mirada a Chip y sonríe. “¡Creo que tendrás que trabajar en el diseño de las páginas Web más pronto de lo que creías!”

Anna y Chip continuaron trabajando en los prototipos diseñando, obteniendo retroalimentación del usuario y modificando el diseño para reflejar los cambios del usuario. Ahora que el trabajo está terminado, tienen una mejor percepción de los requerimientos del sistema.

## EJERCICIOS











Revise los prototipos de informe y de pantalla de los siguientes ejercicios (E-1 a E-10). Registre los cambios en una copia del Formulario de Evaluación de Prototipo. Use Microsoft Access para visualizar los prototipos, después modifique los prototipos de informe y de pantalla con los cambios sugeridos. Imprima los prototipos terminados.


Use los siguientes lineamientos para guiar su análisis:

1. *Alineación de los campos en los informes.* ¿Los campos están alineados correctamente? ¿Los encabezados de columna del informe están alineados correctamente en las columnas? ¿Si el informe tiene títulos a la izquierda de los campos de datos, están alineados correctamente (normalmente a la izquierda)? ¿Los datos están alineados correctamente *dentro* de cada campo de entrada?
2. *Contenido del informe.* ¿El informe contiene todos los datos necesarios? ¿Están los totales y subtotales correctos y útiles? ¿En el informe hay totales extra o datos que no deberían estar? ¿En el informe se imprimen los códigos o el significado de éstos (los códigos se deben evitar debido a que en ocasiones no presentan claramente la información al usuario)?
3. *Revise la apariencia visual del informe.* ¿Su apariencia es agradable? ¿Los campos repetidos están impresos en grupo (es decir, los datos se deben imprimir una sola vez, al principio del grupo)? ¿Hay suficientes líneas en blanco entre los grupos para identificarlos fácilmente?
4. *Alineación de los datos y títulos de la pantalla.* ¿Los títulos están alineados correctamente en las pantallas? ¿Los campos de datos están alineados correctamente? ¿Los datos *dentro* de un campo están alineados correctamente?
5. *Apariencia visual de la pantalla.* ¿La pantalla tiene una apariencia agradable? ¿Hay suficiente espacio vertical entre los campos? ¿Hay suficiente espacio horizontal entre las columnas? ¿Los campos están agrupados lógicamente? ¿Las características, como los botones y casillas de verificación, están agrupadas?

## 6

6. ¿La pantalla contiene todos los elementos funcionales necesarios? Vea si faltan botones que facilitarían el trabajo del usuario con la pantalla; también vea si faltan datos, si hay datos innecesarios o si hay campos que se deben reemplazar con una casilla de verificación o una lista desplegable.

-  E-1. En la figura E6.4 se muestra el listado HARDWARE INVENTORY LISTING. Este listado muestra todas las computadoras personales, clasificadas por campus y salas.
-  E-2. El informe SOFTWARE INVESTMENT REPORT se usa para calcular la cantidad total invertida en el software.
-  E-3. El INFORME DE COMPUTADORAS INSTALADAS muestra la información de las máquinas instaladas.
-  E-4. El prototipo del informe COMPUTER PROBLEM REPORT clasifica todas las máquinas por el costo total de reparaciones e incluye el número de éstas (debido a que algunas máquinas aún tienen garantía, no es tan elevado su costo). Este prototipo se usa para calcular el costo total de las reparaciones en toda la universidad, así como también para identificar las máquinas que tienen problemas.
-  E-5. El informe NEW SOFTWARE INSTALLED REPORT muestra el número de máquinas con cada paquete de software que están instaladas en cada sala de cada campus.
-  E-6. El informe SOFTWARE CROSS-REFERENCE REPORT menciona todas las ubicaciones para cada versión de cada paquete de software.
-  E-7. La pantalla DELETE COMPUTER RECORD se usa para quitar computadoras del sistema. El área de entrada es el campo Hardware Inventory Number. Los otros campos únicamente son de despliegue, para identificar la máquina. A los usuarios les gustaría poder imprimir cada registro antes de que se eliminen. También desean desplazarse a los registros siguientes y anteriores. *Sugerencia:* Examine los campos mostrados en el informe HARDWARE INVENTORY LISTING.
-  E-8. Una pantalla UPDATE MAINTENANCE INFORMATION permite a Mike Crowe cambiar la información de mantenimiento de las computadoras personales. Algunas veces éstos son cambios de rutina, tales como LAST PREVENTIVE MAINTENANCE DATE o NUMBER OF REPAIRS, pero otros cambios podrían ocurrir sólo ocasionalmente, tales como el vencimiento de una garantía. El HARDWARE INVENTORY NUMBER se introduce, y se encuentra el registro COMPUTER RECORD correspondiente. Se despliegan la marca y el modelo para retroalimentación. El operador podría cambiar después los campos WARRANTY, MAINTENANCE INTERVAL, NUMBER OF REPAIRS, LAST PREVENTIVE MAINTENANCE DATE y TOTAL COST OF REPAIRS. A Mike le gustaría imprimir la información de la pantalla, así como también deshacer cualquier cambio, fácilmente.
-  E-9. La SOFTWARE LOCATION INQUIRY despliega la información de las salas y de las máquinas que contienen el software seleccionado. Se introducen la información de TITLE, VERSION NUMBER y OPERATING SYSTEM. Una parte de la información de la pantalla debe mostrar la información sobre CAMPUS LOCATION, ROOM LOCATION, HARDWARE INVENTORY NUMBER, BRAND NAME y MODEL. Los botones permiten al usuario ir al registro siguiente, al registro anterior y cerrar y salir de la pantalla.
-  E-10. La HARDWARE CHARACTERISTICS INQUIRY se usa para localizar máquinas con ciertas características de hardware. El operador introduce un tipo de marca y de unidad DE CD-ROM. Los campos MONITOR y PRINTER CODE tienen listas desplegables para seleccionar los códigos adecuados. La parte desplegada de la pantalla de consulta consiste en CAMPUS, ROOM e INVENTORY NUMBER.

 Los ejercicios precedidos por un icono Web indican que en el sitio Web de este libro hay material con valor agregado.

## 6

01/11/2004

Page 1 of 1

Campus	Room Location	Inventory Number	Brand Name	Model	Present
Central Administration	11111	84004782	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Administration	11111	90875039	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Administration	11111	93955411	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Administration	11111	99381373	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Administration	22222	Total number of machines in room is		4	_____
		10220129	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Administration	99999	Total number of machines in room is		1	_____
Central Administration	99999	22838234	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Administration	99999	24720952	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Administration	99999	33453403	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Administration	99999	34044449	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Administration	99999	40030303	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Administration	99999	47403948	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Administration	99999	56620548	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
		Total number of machines in room is		7	_____
Central Computer Science	22222	Total number of machines at campus is		12	_____
Central Computer Science	22222	34589349	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Computer Science	22222	38376910	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Computer Science	22222	94842282	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Computer Science	99999	Total number of machines in room is		3	_____
		339393	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Zoology	22222	Total number of machines in room is		1	_____
Central Zoology	22222	11398423	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Zoology	22222	28387465	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
Central Zoology	99999	Total number of machines in room is		2	_____
Central Zoology	99999	70722533	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
		99481102	Xxxxxxxxxx	Xxxxxxxxxxxxxxxxxx	_____
		Total number of machines in room is		2	_____
		Total number of machines at campus is		4	_____
		Total number of machines is		24	_____

FIGURA E6.4

Prototipo del listado HARDWARE INVENTORY LISTING. A este informe se le pueden hacer muchas mejoras.