CAPÍTULO

1

EL SOFTWARE Y LA INGENIERÍA DE SOFTWARE

CONCEPTOS CLAVE

actividades estructurales 12
actividades sombrilla. 12
características del software . . . 3
dominios de aplicación 6
ingeniería de software 10
mitos del software 18
práctica 15
principios 16
proceso del software. 12
software heredado 8
webapps 9

Tenía la apariencia clásica de un alto ejecutivo de una compañía importante de software —a la mitad de los 40, con las sienes comenzando a encanecer, esbelto y atlético, con ojos que penetraban al observador mientras hablaba—. Pero lo que dijo me dejó anonadado. "El software ha *muerto*".

Pestañeé con sorpresa y sonreí. "Bromeas, ¿verdad? El mundo es dirigido con software y tu empresa se ha beneficiado mucho de ello. ¡No ha muerto! Está vivo y en desarrollo."

Movió su cabeza de manera enfática. "No, está muerto... al menos como lo conocimos."

Me apoyé en el escritorio. "Continúa."

Habló al tiempo que golpeaba en la mesa con énfasis. "El concepto antiguo del software —lo compras, lo posees y tu trabajo consiste en administrarlo— está llegando a su fin. Hoy día, con Web 2.0 y la computación ubicua cada vez más fuerte, vamos a ver una generación de software por completo diferente. Se distribuirá por internet y se verá exactamente como si estuviera instalado en el equipo de cómputo de cada usuario... pero se encontrará en un servidor remoto."

Tuve que estar de acuerdo. "Entonces, tu vida será mucho más sencilla. Tus muchachos no tendrán que preocuparse por las cinco diferentes versiones de la misma App que utilizan decenas de miles de usuarios."

Sonrió. "Absolutamente. Sólo la versión más reciente estará en nuestros servidores. Cuando hagamos un cambio o corrección, actualizaremos funcionalidad y contenido a cada usuario. Todos lo tendrán en forma instantánea..."

Hice una mueca. "Pero si cometes un error, todos lo tendrán también instantáneamente".

Él se rió entre dientes. "Es verdad, por eso estamos redoblando nuestros esfuerzos para hacer una ingeniería de software aún mejor. El problema es que tenemos que hacerlo 'rápido' porque el mercado se ha acelerado en cada área de aplicación."

UNA MIRADA RÁPIDA

¿Qué es? El software de computadora es el producto que construyen los programadores profesionales y al que después le dan mantenimiento durante un largo tiempo. Incluye programas que se ejecutan en una computadora de cualquier tamaño y arquitectura, contenido que se presenta a medida de que se ejecutan los programas de cómputo e información descriptiva tanto en una copia dura como en formatos virtuales que engloban virtualmente a cualesquiera medios electrónicos. La ingeniería de software está formada por un proceso, un conjunto de métodos (prácticas) y un arreglo de herramientas que permite a los profesionales elaborar software de cómputo de alta calidad.

¿Quién lo hace? Los ingenieros de software elaboran y dan mantenimiento al software, y virtualmente cada persona lo emplea en el mundo industrializado, ya sea en forma directa o indirecta.

¿Por qué es importante? El software es importante porque afecta a casi todos los aspectos de nuestras vidas y ha invadido nuestro comercio, cultura y actividades cotidianas. La ingeniería de software es importante porque nos permite construir sistemas complejos en un tiempo razonable y con alta calidad.

¿Cuáles son los pasos? El software de computadora se construye del mismo modo que cualquier producto exitoso, con la aplicación de un proceso ágil y adaptable para obtener un resultado de mucha calidad, que satisfaga las necesidades de las personas que usarán el producto. En estos pasos se aplica el enfoque de la ingeniería de software.

¿Cuál es el producto final? Desde el punto de vista de un ingeniero de software, el producto final es el conjunto de programas, contenido (datos) y otros productos terminados que constituyen el software de computadora. Pero desde la perspectiva del usuario, el producto final es la información resultante que de algún modo hace mejor al mundo en el que vive.

¿Cómo me aseguro de que lo hice bien? Lea el resto de este libro, seleccione aquellas ideas que sean aplicables al software que usted hace y aplíquelas a su trabajo.

Me recargué en la espalda y coloqué mis manos en mi nuca. "Ya sabes lo que se dice... puedes tenerlo rápido o bien hecho o barato. Escoge dos de estas características..."

"Elijo rápido y bien hecho", dijo mientras comenzaba a levantarse.

También me incorporé. "Entonces realmente necesitas ingeniería de software."

"Ya lo sé", dijo mientras salía. "El problema es que tenemos que llegar a convencer a otra generación más de técnicos de que así es..."

¿Está muerto *realmente* el software? Si lo estuviera, usted no estaría leyendo este libro...

Cita:

"Las ideas y los descubrimientos tecnológicos son los motores que impulsan el crecimiento económico."

Wall Street Journal

El software de computadora sigue siendo la tecnología más importante en la escena mundial. Y también es un ejemplo magnífico de la ley de las consecuencias inesperadas. Hace 50 años, nadie hubiera podido predecir que el software se convertiría en una tecnología indispensable para los negocios, ciencias e ingeniería, ni que permitiría la creación de tecnologías nuevas (por ejemplo, ingeniería genética y nanotecnología), la ampliación de tecnologías ya existentes (telecomunicaciones) y el cambio radical de tecnologías antiguas (la industria de la impresión); tampoco que el software sería la fuerza que impulsaría la revolución de las computadoras personales, que productos de software empacados se comprarían en los supermercados, que el software evolucionaría poco a poco de un producto a un servicio cuando compañías de software "sobre pedido" proporcionaran funcionalidad justo a tiempo a través de un navegador web, que una compañía de software sería más grande y tendría más influencia que casi todas las empresas de la era industrial, que una vasta red llamada internet sería operada con software y evolucionaría y cambiaría todo, desde la investigación en bibliotecas y la compra de productos para el consumidor hasta el discurso político y los hábitos de encuentro de los adultos jóvenes (y no tan jóvenes).

Nadie pudo prever que habría software incrustado en sistemas de toda clase: de transporte, médicos, de telecomunicaciones, militares, industriales, de entretenimiento, en máquinas de oficina... la lista es casi infinita. Y si usted cree en la ley de las consecuencias inesperadas, hay muchos efectos que aún no podemos predecir.

Nadie podía anticipar que millones de programas de computadora tendrían que ser corregidos, adaptados y mejorados a medida que transcurriera el tiempo. Ni que la carga de ejecutar estas actividades de "mantenimiento" absorbería más personas y recursos que todo el trabajo aplicado a la creación de software nuevo.

Conforme ha aumentado la importancia del software, la comunidad de programadores ha tratado continuamente de desarrollar tecnologías que hagan más fácil, rápida y barata la elaboración de programas de cómputo de alta calidad. Algunas de estas tecnologías se dirigen a un dominio específico de aplicaciones (por ejemplo, diseño e implantación de un sitio web), otras se centran en un dominio tecnológico (sistemas orientados a objetos o programación orientada a aspectos), otros más tienen una base amplia (sistemas operativos, como Linux). Sin embargo, todavía falta por desarrollarse una tecnología de software que haga todo esto, y hay pocas probabilidades de que surja una en el futuro. A pesar de ello, las personas basan sus trabajos, confort, seguridad, diversiones, decisiones y sus propias vidas en software de computadora. Más vale que esté bien hecho.

Este libro presenta una estructura que puede ser utilizada por aquellos que hacen software de cómputo —personas que deben hacerlo bien—. La estructura incluye un proceso, un conjunto de métodos y unas herramientas que llamamos *ingeniería de software*.

1.1 La naturaleza del software

En la actualidad, el software tiene un papel dual. Es un producto y al mismo tiempo es el vehículo para entregar un producto. En su forma de producto, brinda el potencial de cómputo incorporado en el hardware de cómputo o, con más amplitud, en una red de computadoras a las

PUNTO CLAVE

El software es tanto un producto como un vehículo para entregar un producto.

que se accede por medio de un hardware local. Ya sea que resida en un teléfono móvil u opere en el interior de una computadora central, el software es un transformador de información —produce, administra, adquiere, modifica, despliega o transmite información que puede ser tan simple como un solo bit o tan compleja como una presentación con multimedios generada a partir de datos obtenidos de decenas de fuentes independientes—. Como vehículo utilizado para distribuir el producto, el software actúa como la base para el control de la computadora (sistemas operativos), para la comunicación de información (redes) y para la creación y control de otros programas (herramientas y ambientes de software).

El software distribuye el producto más importante de nuestro tiempo: *información*. Transforma los datos personales (por ejemplo, las transacciones financieras de un individuo) de modo que puedan ser más útiles en un contexto local, administra la información de negocios para mejorar la competitividad, provee una vía para las redes mundiales de información (la internet) y brinda los medios para obtener información en todas sus formas.

Cita:

"El software es un lugar donde se siembran sueños y se cosechan pesadillas, una ciénega abstracta y mística en la que terribles demonios luchan contra panaceas mágicas, un mundo de hombres lobo y balas de plata."

Brad J. Cox

En el último medio siglo, el papel del software de cómputo ha sufrido un cambio significativo. Las notables mejoras en el funcionamiento del hardware, los profundos cambios en las arquitecturas de computadora, el gran incremento en la memoria y capacidad de almacenamiento, y una amplia variedad de opciones de entradas y salidas exóticas han propiciado la existencia de sistemas basados en computadora más sofisticados y complejos. Cuando un sistema tiene éxito, la sofisticación y complejidad producen resultados deslumbrantes, pero también plantean problemas enormes para aquellos que deben construir sistemas complejos.

En la actualidad, la enorme industria del software se ha convertido en un factor dominante en las economías del mundo industrializado. Equipos de especialistas de software, cada uno centrado en una parte de la tecnología que se requiere para llegar a una aplicación compleja, han reemplazado al programador solitario de los primeros tiempos. A pesar de ello, las preguntas que se hacía aquel programador son las mismas que surgen cuando se construyen sistemas modernos basados en computadora:[1]

- ¿Por qué se requiere tanto tiempo para terminar el software?
- ¿Por qué son tan altos los costos de desarrollo?
- ¿Por qué no podemos detectar todos los errores antes de entregar el software a nuestros clientes?
- ¿Por qué dedicamos tanto tiempo y esfuerzo a mantener los programas existentes?
- ¿Por qué seguimos con dificultades para medir el avance mientras se desarrolla y mantiene el software?

Éstas y muchas otras preguntas, denotan la preocupación sobre el software y la manera en que se desarrolla, preocupación que ha llevado a la adopción de la práctica de la ingeniería del software.

1.1.1 Definición de software

En la actualidad, la mayoría de profesionales y muchos usuarios tienen la fuerte sensación de que entienden el software. Pero, ¿es así?

La descripción que daría un libro de texto sobre software sería más o menos así:

? ¿Cómo se define *software*?

> El software es: 1) instrucciones (programas de cómputo) que cuando se ejecutan proporcionan las características, función y desempeño buscados; 2) estructuras de datos que permiten que los progra-

1 En un excelente libro de ensayos sobre el negocio del software, Tom DeMarco [DeM95] defiende el punto de vista contrario. Dice: "En lugar de preguntar por qué el software cuesta tanto, necesitamos comenzar a preguntar: ¿Qué hemos hecho para hacer posible que el software actual cueste tan poco? La respuesta a esa pregunta nos ayudará a continuar el extraordinario nivel de logro que siempre ha distinguido a la industria del software."

mas manipulen en forma adecuada la información, y 3) información descriptiva tanto en papel como en formas virtuales que describen la operación y uso de los programas.

No hay duda de que podrían darse definiciones más completas.

Pero es probable que una definición más formal no mejore de manera apreciable nuestra comprensión. Para asimilar lo anterior, es importante examinar las características del software que lo hacen diferente de otros objetos que construyen los seres humanos. El software es elemento de un sistema lógico y no de uno físico. Por tanto, tiene características que difieren considerablemente de las del hardware:

1. *El software se desarrolla o modifica con intelecto; no se manufactura en el sentido clásico.*

PUNTO CLAVE
El software se modifica con intelecto, no se manufactura.

Aunque hay algunas similitudes entre el desarrollo de software y la fabricación de hardware, las dos actividades son diferentes en lo fundamental. En ambas, la alta calidad se logra a través de un buen diseño, pero la fase de manufactura del hardware introduce problemas de calidad que no existen (o que se corrigen con facilidad) en el software. Ambas actividades dependen de personas, pero la relación entre los individuos dedicados y el trabajo logrado es diferente por completo (véase el capítulo 24). Las dos actividades requieren la construcción de un "producto", pero los enfoques son distintos. Los costos del software se concentran en la ingeniería. Esto significa que los proyectos de software no pueden administrarse como si fueran proyectos de manufactura.

2. *El software no se "desgasta".*

PUNTO CLAVE
El software no se desgasta, pero sí se deteriora.

La figura 1.1 ilustra la tasa de falla del hardware como función del tiempo. La relación, que es frecuente llamar "curva de tina", indica que el hardware presenta una tasa de fallas relativamente elevada en una etapa temprana de su vida (fallas que con frecuencia son atribuibles a defectos de diseño o manufactura); los defectos se corrigen y la tasa de fallas se abate a un nivel estable (muy bajo, por fortuna) durante cierto tiempo. No obstante, conforme pasa el tiempo, la tasa de fallas aumenta de nuevo a medida que los componentes del hardware resienten los efectos acumulativos de suciedad, vibración, abuso, temperaturas extremas y muchos otros inconvenientes ambientales. En pocas palabras, el hardware comienza a *desgastarse*.

El software no es susceptible a los problemas ambientales que hacen que el hardware se desgaste. Por tanto, en teoría, la curva de la tasa de fallas adopta la forma de la "curva idealizada" que se aprecia en la figura 1.2. Los defectos ocultos ocasionarán ta-

FIGURA 1.1
Curva de fallas del hardware

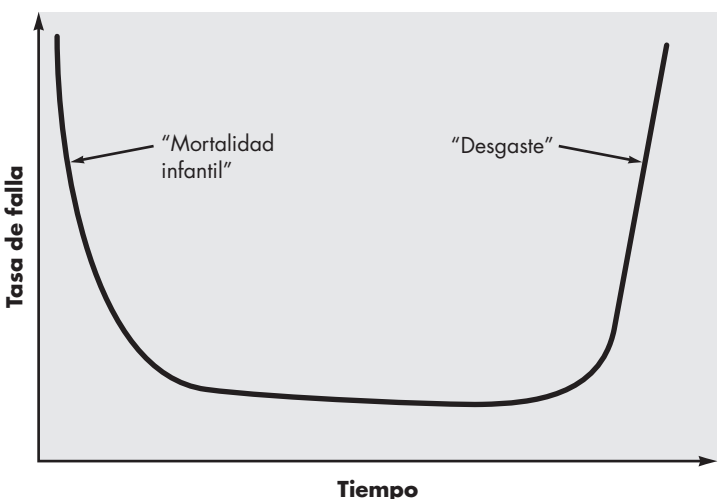
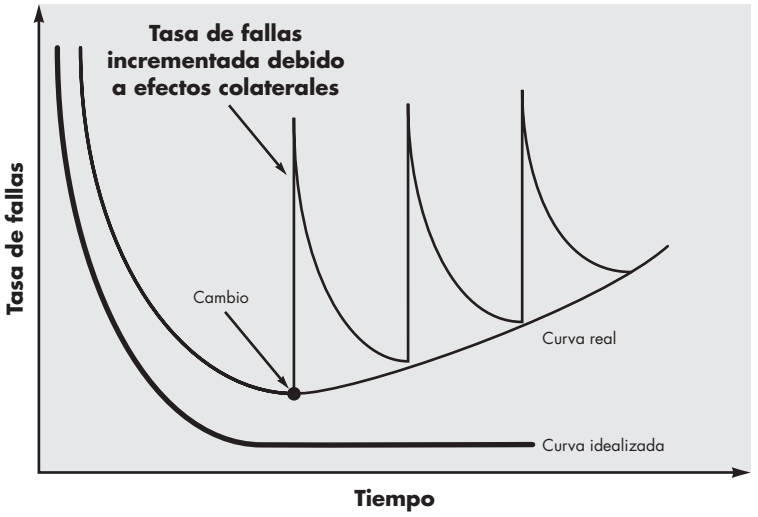


FIGURA 1.2
Curvas de falla del software



CONSEJO

Si quiere reducir el deterioro del software, tendrá que mejorar su diseño (capítulos 8 a 13).

PUNTO CLAVE

Los métodos de la ingeniería de software llevan a reducir la magnitud de los picos y de la pendiente de la curva real en la figura 1.2.

sas elevadas de fallas al comienzo de la vida de un programa. Sin embargo, éstas se corrigen y la curva se aplana, como se indica. La curva idealizada es una gran simplificación de los modelos reales de las fallas del software. Aun así, la implicación está clara: el software no se desgasta, ¡pero sí se *deteriora!*

Esta contradicción aparente se entiende mejor si se considera la curva real en la figura 1.2. Durante su vida,[2] el software sufrirá cambios. Es probable que cuando éstos se realicen, se introduzcan errores que ocasionen que la curva de tasa de fallas tenga aumentos súbitos, como se ilustra en la "curva real" (véase la figura 1.2). Antes de que la curva vuelva a su tasa de fallas original de estado estable, surge la solicitud de otro cambio que hace que la curva se dispare otra vez. Poco a poco, el nivel mínimo de la tasa de fallas comienza a aumentar: el software se está deteriorando como consecuencia del cambio.

Otro aspecto del desgaste ilustra la diferencia entre el hardware y el software. Cuando un componente del hardware se desgasta es sustituido por una refacción. En cambio, no hay refacciones para el software. Cada falla de éste indica un error en el diseño o en el proceso que tradujo el diseño a código ejecutable por la máquina. Entonces, las tareas de mantenimiento del software, que incluyen la satisfacción de peticiones de cambios, involucran una complejidad considerablemente mayor que el mantenimiento del hardware.

Cita:

"Las ideas son los ladrillos con los que se construyen las ideas."

Jason Zebehazy

3. *Aunque la industria se mueve hacia la construcción basada en componentes, la mayor parte del software se construye para un uso individualizado*.

A medida que evoluciona una disciplina de ingeniería, se crea un conjunto de componentes estandarizados para el diseño. Los tornillos estándar y los circuitos integrados preconstruidos son sólo dos de los miles de componentes estándar que utilizan los ingenieros mecánicos y eléctricos conforme diseñan nuevos sistemas. Los componentes reutilizables han sido creados para que el ingeniero pueda concentrarse en los elementos verdaderamente innovadores de un diseño; es decir, en las partes de éste que representan algo nuevo. En el mundo del hardware, volver a usar componentes es una parte

2 En realidad, los distintos participantes solicitan cambios desde el momento en que comienza el desarrollo y mucho antes de que se disponga de la primera versión.

natural del proceso de ingeniería. En el del software, es algo que apenas ha empezado a hacerse a gran escala.

Un componente de software debe diseñarse e implementarse de modo que pueda volverse a usar en muchos programas diferentes. Los modernos componentes reutilizables incorporan tanto los datos como el procesamiento que se les aplica, lo que permite que el ingeniero de software cree nuevas aplicaciones a partir de partes susceptibles de volverse a usar.[3] Por ejemplo, las actuales interfaces interactivas de usuario se construyen con componentes reutilizables que permiten la creación de ventanas gráficas, menús desplegables y una amplia variedad de mecanismos de interacción. Las estructuras de datos y el detalle de procesamiento que se requieren para construir la interfaz están contenidos en una librería de componentes reusables para tal fin.

1.1.2 Dominios de aplicación del software

Actualmente, hay siete grandes categorías de software de computadora que plantean retos continuos a los ingenieros de software:

Software de sistemas: conjunto de programas escritos para dar servicio a otros programas. Determinado software de sistemas (por ejemplo, compiladores, editores y herramientas para administrar archivos) procesa estructuras de información complejas pero deterministas.[4] Otras aplicaciones de sistemas (por ejemplo, componentes de sistemas operativos, manejadores, software de redes, procesadores de telecomunicaciones) procesan sobre todo datos indeterminados. En cualquier caso, el área de software de sistemas se caracteriza por: gran interacción con el hardware de la computadora, uso intensivo por parte de usuarios múltiples, operación concurrente que requiere la secuenciación, recursos compartidos y administración de un proceso sofisticado, estructuras complejas de datos e interfaces externas múltiples.

WebRef

En la dirección **shareware.cnet.com** se encuentra una de las librerías más completas de software compartido y libre.

Software de aplicación: programas aislados que resuelven una necesidad específica de negocios. Las aplicaciones en esta área procesan datos comerciales o técnicos en una forma que facilita las operaciones de negocios o la toma de decisiones administrativas o técnicas. Además de las aplicaciones convencionales de procesamiento de datos, el software de aplicación se usa para controlar funciones de negocios en tiempo real (por ejemplo, procesamiento de transacciones en punto de venta, control de procesos de manufactura en tiempo real).

Software de ingeniería y ciencias: se ha caracterizado por algoritmos "devoradores de números". Las aplicaciones van de la astronomía a la vulcanología, del análisis de tensiones en automóviles a la dinámica orbital del transbordador espacial, y de la biología molecular a la manufactura automatizada. Sin embargo, las aplicaciones modernas dentro del área de la ingeniería y las ciencias están abandonando los algoritmos numéricos convencionales. El diseño asistido por computadora, la simulación de sistemas y otras aplicaciones interactivas, han comenzado a hacerse en tiempo real e incluso han tomado características del software de sistemas.

Software incrustado: reside dentro de un producto o sistema y se usa para implementar y controlar características y funciones para el usuario final y para el sistema en sí. El software incrustado ejecuta funciones limitadas y particulares (por ejemplo, control del tablero de un horno de microondas) o provee una capacidad significativa de funcionamiento y control

3 El desarrollo basado en componentes se estudia en el capítulo 10.

4 El software es *determinista* si es posible predecir el orden y momento de las entradas, el procesamiento y las salidas. El software es *no determinista* si no pueden predecirse el orden y momento en que ocurren éstos.