

# 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*”.

Pestañeeé 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 cotidia-

nas. 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...

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*.

### Cita:

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

Wall Street Journal

## 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.

**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

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.

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:<sup>1</sup>

- ¿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:

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-

**? ¿Cómo se define software?**

<sup>1</sup> 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.*

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".*

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-

**PUNTO CLAVE**

El software se modifica con intelecto, no se manufactura.

**PUNTO CLAVE**

El software no se desgasta, pero sí se deteriora.

**FIGURA 1.1**

Curva de fallas del hardware

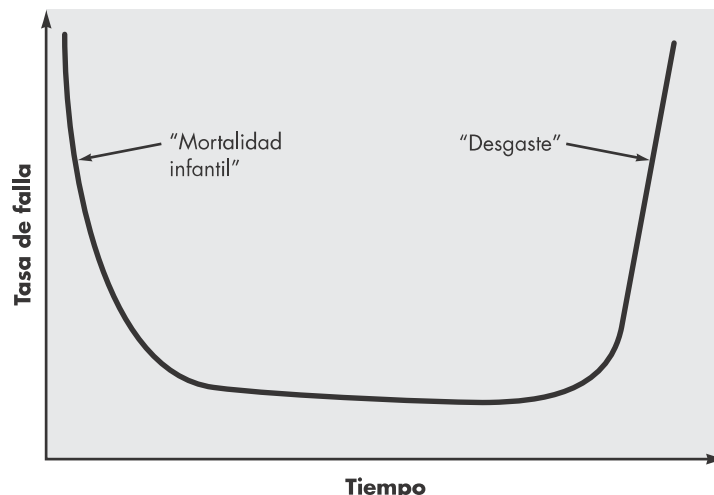
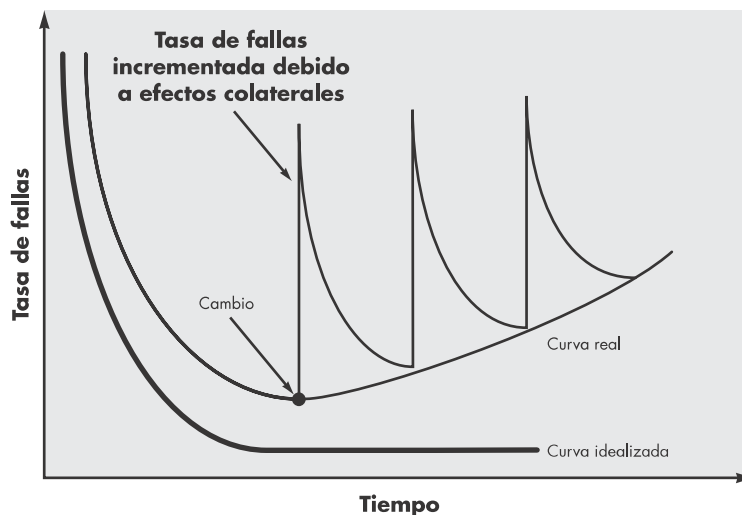


FIGURA 1.2

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.

tas 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,<sup>2</sup> 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.

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

**Cita:**

“Las ideas son los ladrillos con los que se construyen las ideas.”

Jason Zebehazi

<sup>2</sup> 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.<sup>3</sup> Por ejemplo, las actuales interfaces interactivas de usuario se construyen con componentes reutilizables que permiten la creación de ventanas gráficas, menús desplegados 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.<sup>4</sup> 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.

**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

#### WebRef

En la dirección [shareware.cnet.com](http://shareware.cnet.com) se encuentra una de las librerías más completas de software compartido y libre.

<sup>3</sup> El desarrollo basado en componentes se estudia en el capítulo 10.

<sup>4</sup> 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.



(funciones digitales en un automóvil, como el control del combustible, del tablero de control y de los sistemas de frenado).

**Software de línea de productos:** es diseñado para proporcionar una capacidad específica para uso de muchos consumidores diferentes. El software de línea de productos se centra en algún mercado limitado y particular (por ejemplo, control del inventario de productos) o se dirige a mercados masivos de consumidores (procesamiento de textos, hojas de cálculo, gráficas por computadora, multimedia, entretenimiento, administración de base de datos y aplicaciones para finanzas personales o de negocios).

**Aplicaciones web:** llamadas “webapps”, esta categoría de software centrado en redes agrupa una amplia gama de aplicaciones. En su forma más sencilla, las *webapps* son poco más que un conjunto de archivos de hipertexto vinculados que presentan información con uso de texto y gráficas limitadas. Sin embargo, desde que surgió Web 2.0, las *webapps* están evolucionando hacia ambientes de cómputo sofisticados que no sólo proveen características aisladas, funciones de cómputo y contenido para el usuario final, sino que también están integradas con bases de datos corporativas y aplicaciones de negocios.

**Software de inteligencia artificial:** hace uso de algoritmos no numéricos para resolver problemas complejos que no son fáciles de tratar computacionalmente o con el análisis directo. Las aplicaciones en esta área incluyen robótica, sistemas expertos, reconocimiento de patrones (imagen y voz), redes neurales artificiales, demostración de teoremas y juegos.

**Cita:**

“No hay computadora que tenga sentido común.”

Marvin Minsky

Son millones de ingenieros de software en todo el mundo los que trabajan duro en proyectos de software en una o más de estas categorías. En ciertos casos se elaboran sistemas nuevos, pero en muchos otros se corrigen, adaptan y mejoran aplicaciones ya existentes. No es raro que una joven ingeniera de software trabaje en un programa de mayor edad que la de ella... Las generaciones pasadas de los trabajadores del software dejaron un legado en cada una de las categorías mencionadas. Por fortuna, la herencia que dejará la actual generación aligerará la carga de los futuros ingenieros de software. Aun así, nuevos desafíos (capítulo 31) han aparecido en el horizonte.

**Computación en un mundo abierto:** el rápido crecimiento de las redes inalámbricas quizá lleve pronto a la computación verdaderamente ubicua y distribuida. El reto para los ingenieros de software será desarrollar software de sistemas y aplicación que permita a dispositivos móviles, computadoras personales y sistemas empresariales comunicarse a través de redes enormes.

**Construcción de redes:** la red mundial (World Wide Web) se está convirtiendo con rapidez tanto en un motor de computación como en un proveedor de contenido. El desafío para los ingenieros de software es hacer arquitecturas sencillas (por ejemplo, planeación financiera personal y aplicaciones sofisticadas que proporcionen un beneficio a mercados objetivo de usuarios finales en todo el mundo).

**Fuente abierta:** tendencia creciente que da como resultado la distribución de código fuente para aplicaciones de sistemas (por ejemplo, sistemas operativos, bases de datos y ambientes de desarrollo) de modo que mucha gente pueda contribuir a su desarrollo. El desafío para los ingenieros de software es elaborar código fuente que sea autodescriptivo, y también, lo que es más importante, desarrollar técnicas que permitirán tanto a los consumidores como a los desarrolladores saber cuáles son los cambios hechos y cómo se manifiestan dentro del software.

**Cita:**

“No siempre puedes predecir, pero siempre puedes prepararte.”

Anónimo

Es indudable que cada uno de estos nuevos retos obedecerá a la ley de las consecuencias imprevistas y tendrá efectos (para hombres de negocios, ingenieros de software y usuarios finales) que hoy no pueden predecirse. Sin embargo, los ingenieros de software pueden prepararse de-

sarrollando un proceso que sea ágil y suficientemente adaptable para que acepte los cambios profundos en la tecnología y las reglas de los negocios que seguramente surgirán en la década siguiente.

### 1.1.3 Software heredado

Cientos de miles de programas de cómputo caen en uno de los siete dominios amplios de aplicación que se estudiaron en la subsección anterior. Algunos de ellos son software muy nuevo, disponible para ciertos individuos, industria y gobierno. Pero otros programas son más viejos, en ciertos casos *muy* viejos.

Estos programas antiguos —que es frecuente denominar *software heredado*— han sido centro de atención y preocupación continuas desde la década de 1960. Dayani-Fard y sus colegas [Day99] describen el software heredado de la manera siguiente:

Los sistemas de software heredado [...] fueron desarrollados hace varias décadas y han sido modificados de manera continua para que satisfagan los cambios en los requerimientos de los negocios y plataformas de computación. La proliferación de tales sistemas es causa de dolores de cabeza para las organizaciones grandes, a las que resulta costoso mantenerlos y riesgoso hacerlos evolucionar.

Liu y sus colegas [Liu98] amplían esta descripción al hacer notar que “muchos sistemas heredados continúan siendo un apoyo para las funciones básicas del negocio y son ‘indispensables’ para éste”. Además, el software heredado se caracteriza por su longevidad e importancia crítica para el negocio.

Desafortunadamente, en ocasiones hay otra característica presente en el software heredado: *mala calidad*.<sup>5</sup> Hay veces en las que los sistemas heredados tienen diseños que no son susceptibles de extenderse, código confuso, documentación mala o inexistente, casos y resultados de pruebas que nunca se archivaron, una historia de los cambios mal administrada... la lista es muy larga. A pesar de esto, dichos sistemas dan apoyo a las “funciones básicas del negocio y son indispensables para éste”. ¿Qué hacer?

La única respuesta razonable es: *hacer nada*, al menos hasta que el sistema heredado tenga un cambio significativo. Si el software heredado satisface las necesidades de sus usuarios y corre de manera confiable, entonces no falla ni necesita repararse. Sin embargo, conforme pase el tiempo será frecuente que los sistemas de software evolucionen por una o varias de las siguientes razones:

- El software debe adaptarse para que cumpla las necesidades de los nuevos ambientes del cómputo y de la tecnología.
- El software debe ser mejorado para implementar nuevos requerimientos del negocio.
- El software debe ampliarse para que sea operable con otros sistemas o bases de datos modernos.
- La arquitectura del software debe rediseñarse para hacerla viable dentro de un ambiente de redes.

Cuando ocurren estos modos de evolución, debe hacerse la reingeniería del sistema heredado (capítulo 29) para que sea viable en el futuro. La meta de la ingeniería de software moderna es “desarrollar metodologías que se basen en el concepto de evolución; es decir, el concepto de que los sistemas de software cambian continuamente, que los nuevos sistemas de software se

**?** ¿Qué hago si encuentro un sistema heredado de mala calidad?

**?** ¿Qué tipos de cambios se hacen a los sistemas heredados?



**CONSEJO**  
 Todo ingeniero de software debe reconocer que el cambio es natural. No trate de evitarlo.

<sup>5</sup> En este caso, la calidad se juzga con base en el pensamiento moderno de la ingeniería de software, criterio algo injusto, ya que algunos conceptos y principios de la ingeniería de software moderna tal vez no hayan sido bien entendidos en la época en que se desarrolló el software heredado.



desarrollan a partir de los antiguos y [...] que todo debe operar entre sí y cooperar con cada uno de los demás" [Day99].

## 1.2 LA NATURALEZA ÚNICA DE LAS WEBAPPS

### Cita:

"Cuando veamos cualquier tipo de estabilización, la web se habrá convertido en algo completamente diferente."

Louis Monier

En los primeros días de la Red Mundial (entre 1990 y 1995), los *sitios web* consistían en poco más que un conjunto de archivos de hipertexto vinculados que presentaban la información con el empleo de texto y gráficas limitadas. Al pasar el tiempo, el aumento de HTML por medio de herramientas de desarrollo (XML, Java) permitió a los ingenieros de la web brindar capacidad de cómputo junto con contenido de información. Habían nacido los *sistemas y aplicaciones basados en la web*<sup>6</sup> (denominó a éstas en forma colectiva como *webapps*). En la actualidad, las *webapps* se han convertido en herramientas sofisticadas de cómputo que no sólo proporcionan funciones aisladas al usuario final, sino que también se han integrado con bases de datos corporativas y aplicaciones de negocios.

Como se dijo en la sección 1.1.2, las *webapps* son una de varias categorías distintas de software. No obstante, podría argumentarse que las *webapps* son diferentes. Powell [Pow98] sugiere que los sistemas y aplicaciones basados en web "involucran una mezcla entre las publicaciones impresas y el desarrollo de software, entre la mercadotecnia y la computación, entre las comunicaciones internas y las relaciones exteriores, y entre el arte y la tecnología". La gran mayoría de *webapps* presenta los siguientes atributos:

? ¿Qué característica diferencia las *webapps* de otro software?

**Uso intensivo de redes.** Una *webapp* reside en una red y debe atender las necesidades de una comunidad diversa de clientes. La red permite acceso y comunicación mundiales (por ejemplo, internet) o tiene acceso y comunicación limitados (por ejemplo, una intranet corporativa).

**Concurrencia.** A la *webapp* puede acceder un gran número de usuarios a la vez. En muchos casos, los patrones de uso entre los usuarios finales varían mucho.

**Carga impredecible.** El número de usuarios de la *webapp* cambia en varios órdenes de magnitud de un día a otro. El lunes tal vez la utilicen cien personas, el jueves quizá 10 000 usen el sistema.

**Rendimiento.** Si un usuario de la *webapp* debe esperar demasiado (para entrar, para el procesamiento por parte del servidor, para el formado y despliegue del lado del cliente), él o ella quizá decidan irse a otra parte.

**Disponibilidad.** Aunque no es razonable esperar una disponibilidad de 100%, es frecuente que los usuarios de *webapps* populares demanden acceso las 24 horas de los 365 días del año. Los usuarios en Australia o Asia quizá demanden acceso en horas en las que las aplicaciones internas de software tradicionales en Norteamérica no estén en línea por razones de mantenimiento.

**Orientadas a los datos.** La función principal de muchas *webapp* es el uso de hipermedios para presentar al usuario final contenido en forma de texto, gráficas, audio y video. Además, las *webapps* se utilizan en forma común para acceder a información que existe en bases de datos que no son parte integral del ambiente basado en web (por ejemplo, comercio electrónico o aplicaciones financieras).

<sup>6</sup> En el contexto de este libro, el término *aplicación web (webapp)* agrupa todo, desde una simple página web que ayude al consumidor a calcular el pago del arrendamiento de un automóvil hasta un sitio web integral que proporcione servicios completos de viaje para gente de negocios y vacacionistas. En esta categoría se incluyen sitios web completos, funcionalidad especializada dentro de sitios web y aplicaciones de procesamiento de información que residen en internet o en una intranet o extranet.

**Contenido sensible.** La calidad y naturaleza estética del contenido constituye un rasgo importante de la calidad de una *webapp*.

**Evolución continua.** A diferencia del software de aplicación convencional que evoluciona a lo largo de una serie de etapas planeadas y separadas cronológicamente, las aplicaciones web evolucionan en forma continua. No es raro que ciertas *webapp* (específicamente su contenido) se actualicen minuto a minuto o que su contenido se calcule en cada solicitud.

**Inmediatez.** Aunque la *inmediatez* —necesidad apremiante de que el software llegue con rapidez al mercado— es una característica en muchos dominios de aplicación, es frecuente que las *webapps* tengan plazos de algunos días o semanas para llegar al mercado.<sup>7</sup>

**Seguridad.** Debido a que las *webapps* se encuentran disponibles con el acceso a una red, es difícil o imposible limitar la población de usuarios finales que pueden acceder a la aplicación. Con el fin de proteger el contenido sensible y brindar modos seguros de transmisión de los datos, deben implementarse medidas estrictas de seguridad a través de la infraestructura de apoyo de una *webapp* y dentro de la aplicación misma.

**Estética.** Parte innegable del atractivo de una *webapp* es su apariencia y percepción. Cuando se ha diseñado una aplicación para comercializar o vender productos o ideas, la estética tiene tanto que ver con el éxito como el diseño técnico.

Podría argumentarse que otras categorías de aplicaciones estudiadas en la sección 1.1.2 muestran algunos de los atributos mencionados. Sin embargo, las *webapps* casi siempre poseen todos ellos.

### 1.3 INGENIERÍA DE SOFTWARE

Con objeto de elaborar software listo para enfrentar los retos del siglo XXI, el lector debe aceptar algunas realidades sencillas:

#### PUNTO CLAVE

Entender el problema antes de dar una solución.

- El software se ha incrustado profundamente en casi todos los aspectos de nuestras vidas y, como consecuencia, el número de personas que tienen interés en las características y funciones que brinda una aplicación específica<sup>8</sup> ha crecido en forma notable. Cuando ha de construirse una aplicación nueva o sistema incrustado, deben escucharse muchas opiniones. Y en ocasiones parece que cada una de ellas tiene una idea un poco distinta de cuáles características y funciones debiera tener el software. *Se concluye que debe hacerse un esfuerzo concertado para entender el problema antes de desarrollar una aplicación de software.*

#### PUNTO CLAVE

El diseño es una actividad crucial de la ingeniería de software.

- Los requerimientos de la tecnología de la información que demandan los individuos, negocios y gobiernos se hacen más complejos con cada año que pasa. En la actualidad, grandes equipos de personas crean programas de cómputo que antes eran elaborados por un solo individuo. El software sofisticado, que alguna vez se implementó en un ambiente de cómputo predecible y autocontenido, hoy en día se halla incrustado en el interior de todo, desde la electrónica de consumo hasta dispositivos médicos o sistemas de armamento. La complejidad de estos nuevos sistemas y productos basados en computadora demanda atención cuidadosa a las interacciones de todos los elementos del sistema. *Se concluye que el diseño se ha vuelto una actividad crucial.*

<sup>7</sup> Con las herramientas modernas es posible producir páginas web sofisticadas en unas cuantas horas.

<sup>8</sup> En una parte posterior de este libro, llamaré a estas personas “participantes”.