1

Introducción

Objetivos

Los objetivos de este capítulo consisten en introducir al lector a la ingeniería de software y ofrecer un marco conceptual para entender el resto del libro. Al estudiar este capítulo:

- conocerá qué es la ingeniería de software y por qué es importante;
- comprenderá que el desarrollo de diferentes tipos de sistemas de software puede requerir distintas técnicas de ingeniería de software;
- entenderá algunos conflictos éticos y profesionales que son importantes para los ingenieros de software;
- conocerá tres sistemas de diferentes tipos, que se usarán como ejemplos a lo largo del libro.

Contenido

Es imposible operar el mundo moderno sin software. Las infraestructuras nacionales y los servicios públicos se controlan mediante sistemas basados en computadoras, y la mayoría de los productos eléctricos incluyen una computadora y un software de control. La fabricación y la distribución industrial están completamente computarizadas, como el sistema financiero. El entretenimiento, incluida la industria musical, los juegos por computadora, el cine y la televisión, usan software de manera intensiva. Por lo tanto, la ingeniería de software es esencial para el funcionamiento de las sociedades, tanto a nivel nacional como internacional.

Los sistemas de software son abstractos e intangibles. No están restringidos por las propiedades de los materiales, regidos por leyes físicas ni por procesos de fabricación. Esto simplifica la ingeniería de software, pues no existen límites naturales a su potencial. Sin embargo, debido a la falta de restricciones físicas, los sistemas de software pueden volverse rápidamente muy complejos, difíciles de entender y costosos

Hay muchos tipos diferentes de sistemas de software, desde los simples sistemas embebidos, hasta los complejos sistemas de información mundial. No tiene sentido buscar notaciones, métodos o técnicas universales para la ingeniería de software, ya que diferentes tipos de software requieren distintos enfoques. Desarrollar un sistema organizacional de información es completamente diferente de un controlador para un instrumento científico. Ninguno de estos sistemas tiene mucho en común con un juego por computadora de gráficos intensivos. Aunque todas estas aplicaciones necesitan ingeniería de software, no todas requieren las mismas técnicas de ingeniería de software.

Aún existen muchos reportes tanto de proyectos de software que salen mal como de "fallas de software". Por ello, a la ingeniería de software se le considera inadecuada para el desarrollo del software moderno. Sin embargo, desde la perspectiva del autor, muchas de las llamadas fallas del software son consecuencia de dos factores:

Demandas crecientes Conforme las nuevas técnicas de ingeniería de software ayudan a construir sistemas más grandes y complejos, las demandas cambian. Los sistemas tienen que construirse y distribuirse más rápidamente; se requieren sistemas más grandes e incluso más complejos; los sistemas deben tener nuevas capacidades que anteriormente se consideraban imposibles. Los métodos existentes de ingeniería de software no pueden enfrentar la situación, y tienen que desarrollarse nuevas técnicas de ingeniería de software para satisfacer nuevas demandas.

Expectativas bajas Es relativamente sencillo escribir programas de cómputo sin usar métodos y técnicas de ingeniería de software. Muchas compañías se deslizan hacia la ingeniería de software conforme evolucionan sus productos y servicios. No usan métodos de ingeniería de software en su trabajo diario. Por lo tanto, su software con frecuencia es más costoso y menos confiable de lo que debiera. Es necesaria una mejor educación y capacitación en ingeniería de software para solucionar

Los ingenieros de software pueden estar orgullosos de sus logros. Desde luego, todavía se presentan problemas al desarrollar software complejo, pero, sin ingeniería de software, no se habría explorado el espacio ni se tendría Internet o las telecomunicaciones modernas. Todas las formas de viaje serían más peligrosas y caras. La ingeniería de software ha contribuido en gran medida, y sus aportaciones en el siglo XXI serán aún

Historia de la ingeniería de software

El concepto "ingeniería de software" se propuso originalmente en 1968, en una conferencia realizada para discutir lo que entonces se llamaba la "crisis del software" (Naur y Randell, 1969). Se volvió claro que los enfoques individuales al desarrollo de programas no escalaban hacia los grandes y complejos sistemas de software. Éstos no eran confiables, costaban más de lo esperado y se distribuían con demora.

A lo largo de las décadas de 1970 y 1980 se desarrolló una variedad de nuevas técnicas y métodos de ingeniería de software, tales como la programación estructurada, el encubrimiento de información y el desarrollo orientado a objetos. Se perfeccionaron herramientas y notaciones estándar y ahora se usan de manera extensa.

http://www.SoftwareEngineering-9.com/Web/History/

1.1 Desarrollo de software profesional

Muchos individuos escriben programas. En las empresas los empleados hacen programas de hoja de cálculo para simplificar su trabajo; científicos e ingenieros elaboran programas para procesar sus datos experimentales, y los aficionados crean programas para su propio interés y satisfacción. Sin embargo, la gran mayoría del desarrollo de software es una actividad profesional, donde el software se realiza para propósitos de negocios específicos, para su inclusión en otros dispositivos o como productos de software, por ejemplo, sistemas de información, sistemas de CAD, etcétera. El software profesional, destinado a usarse por alguien más aparte de su desarrollador, se lleva a cabo en general por equipos, en vez de individualmente. Se mantiene y cambia a lo largo de su vida.

La ingeniería de software busca apoyar el desarrollo de software profesional, en lugar de la programación individual. Incluye técnicas que apoyan la especificación, el diseño y la evolución del programa, ninguno de los cuales son normalmente relevantes para el desarrollo de software personal. Con el objetivo de ayudarlo a obtener una amplia visión de lo que trata la ingeniería de software, en la figura 1.1 se resumen algunas preguntas planteadas con frecuencia.

Muchos suponen que el software es tan sólo otra palabra para los programas de cómputo. No obstante, cuando se habla de ingeniería de software, esto no sólo se refiere a los programas en sí, sino también a toda la documentación asociada y los datos de configuración requeridos para hacer que estos programas operen de manera correcta. Un sistema de software desarrollado profesionalmente es usualmente más que un solo programa. El sistema por lo regular consta de un número de programas separados y archivos de configuración que se usan para instalar dichos programas. Puede incluir documentación del sistema, que describe la estructura del sistema; documentación del usuario, que explica cómo usar el sistema, y los sitios web para que los usuarios descarguen información reciente del producto.

Ésta es una de las principales diferencias entre el desarrollo de software profesional y el de aficionado. Si usted diseña un programa personal, nadie más lo usará ni tendrá que preocuparse por elaborar guías del programa, documentar el diseño del programa, etcétera. Por el contrario, si crea software que otros usarán y otros ingenieros cambiarán, entonces, en general debe ofrecer información adicional, así como el código del programa.

Pregunta	Respuesta
¿Qué es software?	Programas de cómputo y documentación asociada. Los productos de software se desarrollan para un cliente en particular o para un mercado en general.
¿Cuáles son los atributos del buen software?	El buen software debe entregar al usuario la funcionalidad y el desempeño requeridos, y debe ser sustentable, confiable y utilizable.
¿Qué es ingeniería de software?	La ingeniería de software es una disciplina de la ingeniería que se interesa por todos los aspectos de la producción de software.
	Especificación, desarrollo, validación y evolución del software.
	Las ciencias de la computación se enfocan en teoría y fundamentos; mientras la ingeniería de software se enfoca en el sentido práctico del desarrollo y en la distribución de software.
	La ingeniería de sistemas se interesa por todos los aspectos del desarrollo de sistemas basados en computadoras, incluidos hardware, software e ingeniería de procesos. La ingeniería de software es parte de este proceso más general.
	Se enfrentan con una diversidad creciente, demandas por tiempos de distribución limitados y desarrollo de software confiable.
	Aproximadamente 60% de los costos del software son de desarrollo, y 40% de prueba. Para el software elaborado específicamente, los costos de evolución superan con frecuencia los costos de desarrollo.
	Aun cuando todos los proyectos de software deben gestionarse y desarrollarse de manera profesional, existen diferentes técnicas que son adecuadas para distintos tipos de sistema. Por ejemplo, los juegos siempre deben diseñarse usando una serie de prototipos, mientras que los sistemas críticos de control de seguridad requieren de una especificación completa y analizable para su desarrollo. Por lo tanto, no puede decirse que un método sea mejor que otro.
	La Web ha llevado a la disponibilidad de servicios de software y a la posibilidad de desarrollar sistemas basados en servicios distribuidos ampliamente. El desarrollo de sistemas basados en Web ha conducido a importantes avances en lenguajes de programación y reutilización de software.

Los ingenieros de software están interesados por el desarrollo de productos de software (es decir, software que puede venderse a un cliente). Existen dos tipos de productos de software:

Productos genéricos Consisten en sistemas independientes que se producen por una organización de desarrollo y se venden en el mercado abierto a cualquier cliente

que desee comprarlos. Ejemplos de este tipo de productos incluyen software para PC, como bases de datos, procesadores de texto, paquetes de dibujo y herramientas de administración de proyectos. También abarcan las llamadas aplicaciones verticales diseñadas para cierto propósito específico, tales como sistemas de información de librería, sistemas de contabilidad o sistemas para mantener registros dentales.

2. *Productos personalizados (o a la medida)* Son sistemas que están destinados para un cliente en particular. Un contratista de software desarrolla el programa especialmente para dicho cliente. Ejemplos de este tipo de software incluyen los sistemas de control para dispositivos electrónicos, sistemas escritos para apoyar cierto proceso empresarial y los sistemas de control de tráfico aéreo.

Una diferencia importante entre estos tipos de software es que, en productos genéricos, la organización que desarrolla el software controla la especificación del mismo. Para los productos personalizados, la organización que compra el software generalmente desarrolla y controla la especificación, por lo que los desarrolladores de software deben trabajar siguiendo dicha especificación.

Sin embargo, la distinción entre estos tipos de producto de sistemas se vuelve cada vez más difusa. Ahora, cada vez más sistemas se construyen con un producto genérico como base, que luego se adapta para ajustarse a los requerimientos de un cliente. Los sistemas Enterprise Resource Planning (ERP, planeación de recursos empresariales), como el sistema SAP, son los mejores ejemplos de este enfoque. Aquí, un sistema grande y complejo se adapta a una compañía al incorporar la información acerca de las reglas y los procesos empresariales, los reportes requeridos, etcétera.

Cuando se habla de la calidad del software profesional, se debe considerar que el software lo usan y cambian personas, además de sus desarrolladores. En consecuencia, la calidad no tiene que ver sólo con lo que hace el software. En cambio, debe incluir el comportamiento del software mientras se ejecuta, y la estructura y organización de los programas del sistema y la documentación asociada. Esto se refleja en los llamados calidad o atributos no funcionales del software. Ejemplos de dichos atributos son el tiempo de respuesta del software ante la duda de un usuario y la comprensibilidad del código del programa.

El conjunto específico de atributos que se espera de un sistema de software depende evidentemente de su aplicación. Así, un sistema bancario debe ser seguro, un juego interactivo debe tener capacidad de respuesta, un sistema de conmutación telefónica debe ser confiable, etcétera. Esto puede generalizarse en el conjunto de atributos que se muestra en la figura 1.2, los cuales consideran las características esenciales de un sistema de software profesional.

1.1.1 Ingeniería de software

La ingeniería de software es una disciplina de ingeniería que se interesa por todos los aspectos de la producción de software, desde las primeras etapas de la especificación del sistema hasta el mantenimiento del sistema después de que se pone en operación. En esta definición se presentan dos frases clave:

1. *Disciplina de ingeniería* Los ingenieros hacen que las cosas funcionen. Aplican teorías, métodos y herramientas donde es adecuado. Sin embargo, los usan de manera

	Descripción
Mantenimiento	El software debe escribirse de tal forma que pueda evolucionar para satisfacer las necesidades cambiantes de los clientes. Éste es un atributo crítico porque el cambio del software es un requerimiento inevitable de un entorno empresarial variable.
Confiabilidad y seguridad	La confiabilidad del software incluye un rango de características que abarcan fiabilidad, seguridad y protección. El software confiable no tiene que causar daño físico ni económico, en caso de falla del sistema. Los usuarios malintencionados no deben tener posibilidad de acceder al sistema o dañarlo.
Eficiencia	El software no tiene que desperdiciar los recursos del sistema, como la memoria y los ciclos del procesador. Por lo tanto, la eficiencia incluye capacidad de respuesta, tiempo de procesamiento, utilización de memoria, etcétera.
	El software debe ser aceptable al tipo de usuarios para quienes se diseña. Esto significa que necesita ser comprensible, utilizable y compatible con otros sistemas que ellos usan.

selectiva y siempre tratan de encontrar soluciones a problemas, incluso cuando no hay teorías ni métodos aplicables. Los ingenieros también reconocen que deben trabajar ante restricciones organizacionales y financieras, de modo que buscan soluciones dentro de tales limitaciones.

Todos los aspectos de la producción del software La ingeniería de software no sólo se interesa por los procesos técnicos del desarrollo de software, sino también incluye actividades como la administración del proyecto de software y el desarrollo de herramientas, así como métodos y teorías para apoyar la producción de

La ingeniería busca obtener resultados de la calidad requerida dentro de la fecha y del presupuesto. A menudo esto requiere contraer compromisos: los ingenieros no deben ser perfeccionistas. Sin embargo, las personas que diseñan programas para sí mismas podrían pasar tanto tiempo como deseen en el desarrollo del programa.

En general, los ingenieros de software adoptan en su trabajo un enfoque sistemático y organizado, pues usualmente ésta es la forma más efectiva de producir software de alta calidad. No obstante, la ingeniería busca seleccionar el método más adecuado para un conjunto de circunstancias y, de esta manera, un acercamiento al desarrollo más creativo y menos formal sería efectivo en ciertas situaciones. El desarrollo menos formal es particularmente adecuado para la creación de sistemas basados en la Web, que requieren una mezcla de habilidades de software y diseño gráfico.

La ingeniería de software es importante por dos razones:

Cada vez con mayor frecuencia, los individuos y la sociedad se apoyan en los avanzados sistemas de software. Por ende, se requiere producir económica y rápidamente sistemas confiables.

2. A menudo resulta más barato a largo plazo usar métodos y técnicas de ingeniería de software para los sistemas de software, que sólo diseñar los programas como si fuera un proyecto de programación personal. Para muchos tipos de sistemas, la mayoría de los costos consisten en cambiar el software después de ponerlo en operación.

El enfoque sistemático que se usa en la ingeniería de software se conoce en ocasiones como proceso de software. Un proceso de software es una secuencia de actividades que conducen a la elaboración de un producto de software. Existen cuatro actividades fundamentales que son comunes a todos los procesos de software, y éstas son:

1. Especificación del software, donde clientes e ingenieros definen el software que se producirá y las restricciones en su operación.
2. Desarrollo del software, donde se diseña y programa el software.
3. Validación del software, donde se verifica el software para asegurar que sea lo que el cliente requiere.
4. Evolución del software, donde se modifica el software para reflejar los requerimientos cambiantes del cliente y del mercado.

Diferentes tipos de sistemas necesitan distintos procesos de desarrollo. Por ejemplo, el software en tiempo real en una aeronave debe especificarse por completo antes de comenzar el desarrollo. En los sistemas de comercio electrónico, la especificación y el programa por lo general se desarrollan en conjunto. En consecuencia, tales actividades genéricas pueden organizarse en diferentes formas y describirse en distintos niveles de detalle, dependiendo del tipo de software que se vaya a desarrollar. En el capítulo 2 se describen con más puntualidad los procesos de software.

La ingeniería de software se relaciona con las ciencias de la computación y la ingeniería de sistemas:

1. Las ciencias de la computación se interesan por las teorías y los métodos que subyacen en las computadoras y los sistemas de software, en tanto que la ingeniería de software se preocupa por los asuntos prácticos de la producción del software. Cierto conocimiento de ciencias de la computación es esencial para los ingenieros de software, del mismo modo que cierto conocimiento de física lo es para los ingenieros electricistas. Sin embargo, con frecuencia la teoría de las ciencias de la computación es más aplicable a programas relativamente pequeños. Las teorías de las ciencias de la computación no siempre pueden aplicarse a grandes problemas complejos que requieren una solución de software.
2. La ingeniería de sistemas se interesa por todos los aspectos del desarrollo y la evolución de complejos sistemas, donde el software tiene un papel principal. Por lo tanto, la ingeniería de sistemas se preocupa por el desarrollo de hardware, el diseño de políticas y procesos, la implementación del sistema, así como por la ingeniería de software. Los ingenieros de sistemas intervienen en la especificación del sistema, definiendo su arquitectura global y, luego, integrando las diferentes partes para crear el sistema terminado. Están menos preocupados por la ingeniería de los componentes del sistema (hardware, software, etcétera).

Como se expone en la siguiente sección, hay muchos tipos diferentes de software. No existe un método o una técnica universales en la ingeniería de software que sea aplicable para todos éstos. No obstante, tres problemas generales afectan a muy diversos tipos de software:

Cada vez con mayor frecuencia se requieren sistemas que operen como distribuidos a través de redes que incluyan diferentes tipos de computadoras y dispositivos móviles. Es posible que el software se ejecute tanto en computadoras de propósito general como en teléfonos móviles. Se tendrá que integrar con frecuencia el nuevo software con sistemas legados más viejos, escritos en diferentes lenguajes de programación. El reto aquí es desarrollar técnicas para construir software confiable que sea suficientemente flexible para enfrentar esa heterogeneidad.

Cambio empresarial y social Los negocios y la sociedad cambian de manera increíblemente rápida, conforme se desarrollan las economías emergentes y nuevas tecnologías están a la disposición. Ambos necesitan tener la posibilidad de cambiar su software existente y desarrollar rápidamente uno nuevo. Muchas técnicas tradicionales de la ingeniería de software consumen tiempo, y generalmente la entrega de los nuevos sistemas tarda más de lo planeado. Requieren evolucionar de modo que se reduzca el tiempo necesario para que el software dé valor a sus clientes.

Seguridad y confianza Dado que el software está vinculado con todos los aspectos de la vida, es esencial confiar en dicho software. Esto es especialmente cierto para los sistemas de software remoto a los que se accede a través de una página Web o una interfaz de servicio Web. Es necesario asegurarse de que usuarios malintencionados no puedan atacar el software y que se conserve la seguridad de la información.

Desde luego, éstos no son problemas independientes. Por ejemplo, quizá sea necesario realizar cambios rápidos a un sistema legado con la finalidad de dotarlo con una interfaz de servicio Web. Para enfrentar dichos retos se necesitarán nuevas herramientas y técnicas, así como formas innovadoras de combinar y usar los métodos existentes de ingeniería de software.

1.1.2 Diversidad de la ingeniería de software

La ingeniería de software es un enfoque sistemático para la producción de software que toma en cuenta los temas prácticos de costo, fecha y confiabilidad, así como las necesidades de clientes y fabricantes de software. Como este enfoque sistemático realmente implementado varía de manera drástica dependiendo de la organización que desarrolla el software, el tipo de software y los individuos que intervienen en el proceso de desarrollo, no existen métodos y técnicas universales de ingeniería de software que sean adecuados para todos los sistemas y las compañías. Más bien, durante los últimos 50 años evolucionó un conjunto de métodos y herramientas de ingeniería de software.

Quizás el factor más significativo en la determinación de qué métodos y técnicas de la ingeniería de software son más importantes, es el tipo de aplicación que está siendo desarrollada. Existen muchos diferentes tipos de aplicación, incluidos los siguientes:

Aplicaciones independientes Se trata de sistemas de aplicación que corren en una computadora local, como una PC, e incluyen toda la funcionalidad necesaria

y no requieren conectarse a una red. Ejemplos de tales aplicaciones son las de oficina en una PC, programas CAD, software de manipulación de fotografías, etcétera.

2. *Aplicaciones interactivas basadas en transacción* Consisten en aplicaciones que se ejecutan en una computadora remota y a las que los usuarios acceden desde sus propias PC o terminales. Evidentemente, en ellas se incluyen aplicaciones Web como las de comercio electrónico, donde es posible interactuar con un sistema remoto para comprar bienes y servicios. Esta clase de aplicación también incluye sistemas empresariales, donde una organización brinda acceso a sus sistemas a través de un navegador Web o un programa de cliente de propósito específico y servicios basados en nube, como correo electrónico y compartición de fotografías. Las aplicaciones interactivas incorporan con frecuencia un gran almacén de datos al que se accede y actualiza en cada transacción.

3. *Sistemas de control embebido* Se trata de sistemas de control de software que regulan y gestionan dispositivos de hardware. Numéricamente, quizás existen más sistemas embebidos que cualquier otro tipo de sistema. Algunos ejemplos de sistemas embebidos incluyen el software en un teléfono móvil (celular), el software que controla los frenos antibloqueo de un automóvil y el software en un horno de microondas para controlar el proceso de cocinado.

4. *Sistemas de procesamiento en lotes* Son sistemas empresariales que se diseñan para procesar datos en grandes lotes (batch). Procesan gran cantidad de entradas individuales para crear salidas correspondientes. Los ejemplos de sistemas batch incluyen sistemas de facturación periódica, como los sistemas de facturación telefónica y los sistemas de pago de salario.

5. *Sistemas de entretenimiento* Son sistemas para uso sobre todo personal, que tienen la intención de entretener al usuario. La mayoría de estos sistemas son juegos de uno u otro tipo. La calidad de interacción ofrecida al usuario es la característica más importante de los sistemas de entretenimiento.

6. *Sistemas para modelado y simulación* Éstos son sistemas que desarrollan científicos e ingenieros para modelar procesos o situaciones físicas, que incluyen muchos objetos separados interactuantes. Dichos sistemas a menudo son computacionalmente intensivos y para su ejecución requieren sistemas paralelos de alto desempeño.

7. *Sistemas de adquisición de datos* Son sistemas que desde su entorno recopilan datos usando un conjunto de sensores, y envían dichos datos para su procesamiento a otros sistemas. El software tiene que interactuar con los sensores y se instala regularmente en un ambiente hostil, como en el interior de un motor o en una ubicación remota.

8. *Sistemas de sistemas* Son sistemas compuestos de un cierto número de sistemas de software. Algunos de ellos son producto del software genérico, como un programa de hoja de cálculo. Otros sistemas en el ensamble pueden estar especialmente escritos para ese entorno.

Desde luego, los límites entre estos tipos de sistemas son difusos. Si se desarrolla un juego para un teléfono móvil (celular), se debe tomar en cuenta las mismas restricciones (energía, interacción de hardware) que las de los desarrolladores del software del

teléfono. Los sistemas de procesamiento por lotes se usan con frecuencia en conjunción con sistemas basados en la Web. Por ejemplo, en una compañía, las solicitudes de gastos de viaje se envían mediante una aplicación Web, aunque se procesa en una aplicación batch para pago mensual.

Para cada tipo de sistema se usan distintas técnicas de ingeniería de software, porque el software tiene características muy diferentes. Por ejemplo, un sistema de control embebido en un automóvil es crítico para la seguridad y se quema en la ROM cuando se instala en el vehículo; por consiguiente, es muy costoso cambiarlo. Tal sistema necesita verificación y validación muy exhaustivas, de tal modo que se minimicen las probabilidades de volver a llamar para revisión a automóviles, después de su venta, para corregir los problemas del software. La interacción del usuario es mínima (o quizás inexistente), por lo que no hay necesidad de usar un proceso de desarrollo que se apoye en el prototipo de interfaz de usuario.

Para un sistema basado en la Web sería adecuado un enfoque basado en el desarrollo y la entrega iterativos, con un sistema de componentes reutilizables. Sin embargo, tal enfoque podría no ser práctico para un sistema de sistemas, donde tienen que definirse por adelantado las especificaciones detalladas de las interacciones del sistema, de modo que cada sistema se desarrolle por separado.

No obstante, existen fundamentos de ingeniería de software que se aplican a todos los tipos de sistema de software:

Deben llevarse a cabo usando un proceso de desarrollo administrado y comprendido. La organización que diseña el software necesita planear el proceso de desarrollo, así como tener ideas claras acerca de lo que producirá y el tiempo en que estará completado. Desde luego, se usan diferentes procesos para distintos tipos de software.

La confiabilidad y el desempeño son importantes para todos los tipos de sistemas. El software tiene que comportarse como se espera, sin fallas, y cuando se requiera estar disponible. Debe ser seguro en su operación y, tanto como sea posible, también contra ataques externos. El sistema tiene que desempeñarse de manera eficiente y no desperdiciar recursos.

Es importante comprender y gestionar la especificación y los requerimientos del software (lo que el software debe hacer). Debe conocerse qué esperan de él los diferentes clientes y usuarios del sistema, y gestionar sus expectativas, para entregar un sistema útil dentro de la fecha y presupuesto.

Tiene que usar de manera tan efectiva como sea posible los recursos existentes. Esto significa que, donde sea adecuado, hay que reutilizar el software que se haya desarrollado, en vez de diseñar uno nuevo.

Estas nociones fundamentales sobre proceso, confiabilidad, requerimientos, gestión y reutilización, son temas importantes de este libro. Diferentes métodos los reflejan de formas diversas, pero subyacen en todo el desarrollo de software profesional.

Hay que destacar que estos fundamentos no cubren la implementación ni la programación. En este libro no se estudian técnicas específicas de programación, ya que ellas varían drásticamente de un tipo de sistema a otro. Por ejemplo, un lenguaje de guiones (scripts), como Ruby, sirve para programación de sistemas basados en la Web, aunque sería totalmente inadecuado para ingeniería de sistemas embebidos.

1.1.3 Ingeniería de software y la Web

El desarrollo de la World Wide Web tuvo un profundo efecto en todas nuestras vidas. En un inicio, la Web fue básicamente un almacén de información universal accesible que tuvo escaso efecto sobre los sistemas de software. Dichos sistemas corrían en computadoras locales y eran sólo accesibles desde el interior de una organización. Alrededor del año 2000, la Web comenzó a evolucionar, y a los navegadores se les agregaron cada vez más funcionalidades. Esto significó que los sistemas basados en la Web podían desarrollarse donde se tuviera acceso a dichos sistemas usando un navegador Web, en lugar de una interfaz de usuario de propósito específico. Esta situación condujo al desarrollo de una gran variedad de nuevos productos de sistemas que entregaban servicios innovadores, a los cuales se ingresaba desde la Web. A menudo los financiaban los anuncios publicitarios que se desplegaban en la pantalla del usuario y no requerían del pago directo de los usuarios.

Así como estos productos de sistemas, el desarrollo de navegadores Web que corrieran pequeños programas y realizaran cierto procesamiento local condujo a una evolución en los negocios y el software organizacional. En lugar de elaborar software e implementarlo en las PC de los usuarios, el software se implementaba en un servidor Web. Este avance hizo mucho más barato cambiar y actualizar el software, pues no había necesidad de instalar el software en cada PC. También redujo costos, ya que el desarrollo de interfaces de usuario es bastante caro. En consecuencia, dondequiera que fuera posible hacerlo, muchos negocios se mudaron a la interacción basada en la Web con sistemas de software de la compañía.

La siguiente etapa en el desarrollo de los sistemas basados en la Web fue la noción de los servicios Web. Estos últimos son componentes de software que entregan funcionalidad específica y útil, y a los que se accede desde la Web. Las aplicaciones se construyen al integrar dichos servicios Web que ofrecen diferentes compañías. En principio, esta vinculación suele ser dinámica, de modo que se utilice una aplicación cada vez que se ejecutan diferentes servicios Web. En el capítulo 19 se analiza este acercamiento al desarrollo del software.

En los últimos años se desarrolló la noción de "software como servicio". Se propuso que el software no correría usualmente en computadoras locales, sino en "nubes de cómputo" a las que se accede a través de Internet. Si usted utiliza un servicio como el correo basado en la Web, usa un sistema basado en nube. Una nube de computación es un enorme número de sistemas de cómputo vinculados que comparten muchos usuarios. Éstos no compran software, sino que pagan según el tiempo de software que se utiliza, o también se les otorga acceso gratuito a cambio de ver anuncios publicitarios que se despliegan en sus pantallas.

Por consiguiente, la llegada de la Web condujo a un significativo cambio en la forma en que se organiza el software empresarial. Antes de la Web, las aplicaciones empresariales eran básicamente monolíticas, los programas corrían en computadoras individuales o en grupos de computadoras. Las comunicaciones eran locales dentro de una organización. Ahora el software está ampliamente distribuido, en ocasiones a lo largo del mundo. Las aplicaciones empresariales no se programan desde cero, sino que requieren la reutilización extensiva de componentes y programas.

En efecto, este cambio radical en la organización del software tuvo que conducir a modificaciones en las formas en que los sistemas basados en la Web se someten a ingeniería. Por ejemplo:

1. La reutilización de software se ha convertido en el enfoque dominante para construir sistemas basados en la Web. Cuando se construyen tales sistemas, uno piensa en cómo ensamblarlos a partir de componentes y sistemas de software preexistentes.

Ahora se reconoce en general que no es práctico especificar por adelantado todos los requerimientos para tales sistemas. Los sistemas basados en la Web deben desarrollarse y entregarse de manera progresiva.

Las interfaces de usuario están restringidas por las capacidades de los navegadores Web. Aunque tecnologías como AJAX (Holdener, 2008) significan que es posible crear valiosas interfaces dentro de un navegador Web, dichas tecnologías aún son difíciles de emplear. Se usan más comúnmente los formatos Web con escritura de guiones local. Las interfaces de aplicación en sistemas basados en la Web con frecuencia son más deficientes que las interfaces de usuario específicamente diseñadas en productos de sistema PC.

Las ideas fundamentales de la ingeniería de software, discutidas en la sección anterior, se aplican en el software basado en la Web de la misma forma que en otros tipos de sistemas de software. En el siglo XX, la experiencia obtenida con el desarrollo de grandes sistemas todavía es relevante para el software basado en la Web.

Ética en la ingeniería de software

Como otras disciplinas de ingeniería, la ingeniería de software se realiza dentro de un marco social y legal que limita la libertad de la gente que trabaja en dicha área. Como ingeniero de software, usted debe aceptar que su labor implica responsabilidades mayores que la simple aplicación de habilidades técnicas. También debe comportarse de forma ética y moralmente responsable para ser respetado como un ingeniero pro-

No sobra decir que debe mantener estándares normales de honestidad e integridad. No debe usar sus habilidades y experiencia para comportarse de forma deshonesta o de un modo que desacredite la profesión de ingeniería de software. Sin embargo, existen áreas donde los estándares de comportamiento aceptable no están acotados por la legislación, sino por la noción más difusa de responsabilidad profesional. Algunas de ellas

Confidencialidad Por lo general, debe respetar la confidencialidad de sus empleadores o clientes sin importar si se firmó o no un acuerdo formal sobre la misma.

No debe desvirtuar su nivel de competencia. Es decir, no hay que aceptar de manera intencional trabajo que esté fuera de su competencia.

Derechos de propiedad intelectual Tiene que conocer las leyes locales que rigen el uso de la propiedad intelectual, como las patentes y el *copyright*. Debe ser cuidadoso para garantizar que se protege la propiedad intelectual de empleadores y clientes.

Mal uso de computadoras No debe emplear sus habilidades técnicas para usar incorrectamente las computadoras de otros individuos. El mal uso de computadoras varía desde lo relativamente trivial (esto es, distraerse con los juegos de la PC del compañero) hasta lo extremadamente serio (diseminación de virus u otro

Código de ética y práctica profesional de la ingeniería de software

ACM/IEEE-CS Fuerza de trabajo conjunta acerca de ética y prácticas profesionales de la ingeniería de software

PREÁMBULO

La versión corta del código resume las aspiraciones a un alto nivel de abstracción; las cláusulas que se incluyen en la versión completa dan ejemplos y detalles de cómo dichas aspiraciones cambian la forma en que actuamos como profesionales de la ingeniería de software. Sin las aspiraciones, los detalles pueden volverse legalistas y tediosos; mientras que sin los detalles, las aspiraciones suelen volverse muy resonantes pero vacías; en conjunto, aspiraciones y detalles forman un código cohesivo.

Los ingenieros de software deben comprometerse a hacer del análisis, la especificación, el diseño, el desarrollo, la prueba y el mantenimiento del software, una profesión benéfica y respetada. De acuerdo con su compromiso con la salud, la seguridad y el bienestar del público, los ingenieros de software tienen que adherirse a los ocho principios siguientes:

1. PÚBLICO: Los ingenieros de software deben actuar consecuentemente con el interés del público.
2. CLIENTE Y EMPLEADOR: Los ingenieros de software tienen que comportarse de tal forma que fomente el mejor interés para su cliente y empleador, en coherencia con el interés público.
3. PRODUCTO: Los ingenieros de software deben garantizar que sus productos y modificaciones relacionadas satisfagan los estándares profesionales más altos posibles.
4. JUICIO: Los ingenieros de software tienen que mantener integridad e independencia en su juicio profesional.
5. GESTIÓN: Los administradores y líderes en la ingeniería de software deben suscribir y promover un enfoque ético a la gestión del desarrollo y el mantenimiento del software.
6. PROFESIÓN: Los ingenieros de software tienen que fomentar la integridad y la reputación de la profesión consecuente con el interés público.
7. COLEGAS: Los ingenieros de software deben ser justos con sus colegas y apoyarlos.
8. UNO MISMO: Los ingenieros de software tienen que intervenir en el aprendizaje para toda la vida, en cuanto a la práctica de su profesión, y promover un enfoque ético.

Figura 1.3 El código de ética ACM/IEEE (© IEEE/ACM, 1999)

Las sociedades e instituciones profesionales tienen un importante papel que desempeñar en el establecimiento de estándares éticos. Organizaciones como la ACM, el instituto de ingenieros eléctricos y electrónicos (IEEE) y la British Computer Society publican un código de conducta profesional o código de ética. Los integrantes de tales organizaciones se comprometen a seguir dicho código cuando firman al afiliarse. Estos códigos de conducta se preocupan en general por el comportamiento ético fundamental.

Las asociaciones profesionales, sobre todo la ACM y el IEEE, han cooperado para elaborar conjuntamente un código de ética y práctica profesionales. Este código existe tanto de manera simplificada (figura 1.3) como pormenorizada (Gotterbarn *et al.*, 1999) que agrega detalle y sustancia a la versión más corta. Los fundamentos detrás de este código se resumen en los primeros dos párrafos de la forma pormenorizada:

> *Las computadoras tienen una función central y creciente en el comercio, la industria, el gobierno, la medicina, la educación, el entretenimiento y la sociedad en general. Los ingenieros de software son quienes contribuyen, mediante la participación directa o con la enseñanza, al análisis, la especificación, el diseño, el desarrollo, la certificación, el mantenimiento y la prueba de los sistemas de software.*

Debido a su función en el desarrollo de los sistemas de software, los ingenieros de software tienen oportunidades significativas para hacer lo correcto o causar daño, para permitir que otros hagan lo correcto o causen daño, o para influir en otros para hacer lo correcto o causar daño. Para garantizar, tanto como sea posible, que sus esfuerzos serán usados correctamente, los ingenieros de software deben comprometerse a hacer de la ingeniería de software una profesión benéfica y respetada. En concordancia con dicho compromiso, los ingenieros de software tienen que adherirse al siguiente Código de Ética y Práctica Profesional.

El código contiene ocho principios relacionados con el comportamiento y las decisiones tomadas por ingenieros de software profesionales, incluidos practicantes, educadores, administradores, supervisores y políticos, así como por aprendices y estudiantes de la profesión. Los principios identifican las relaciones éticamente responsables en las que participan individuos, grupos y organizaciones, así como las obligaciones principales dentro de estas relaciones. Las cláusulas de cada principio son ilustraciones de algunas de las obligaciones incluidas en dichas relaciones. Tales obligaciones se fundamentan en el sentido humano del ingeniero de software, en el cuidado especial que se debe a las personas afectadas por el trabajo de los ingenieros de software, y los elementos únicos de la práctica de la ingeniería de software. El código las formula como obligaciones de quienquiera que afirme o aspire a ser ingeniero de software.

En cualquier situación donde distintos individuos tengan diferentes visiones y objetivos, es probable que usted enfrente dilemas éticos. Por ejemplo, si está en desacuerdo, en principio, con las políticas de los ejecutivos de más alto nivel en la compañía, ¿cómo reaccionaría? Claramente, esto depende de cada individuo y de la naturaleza de la discrepancia. ¿Es mejor argumentar un caso para su posición desde el interior de la organización o renunciar en principio? Si siente que existen problemas con un proyecto de software, ¿cuándo los reporta a la administración? Si los discute mientras apenas son un indicio, puede estar exagerando su reacción ante una situación; y si los deja para más tarde, quizá sea ya imposible resolver las dificultades.

Estos dilemas éticos los enfrentamos todos en la vida profesional y, por fortuna, en la mayoría de los casos son relativamente menores o pueden resolverse sin demasiada dificultad. En caso de que no puedan solucionarse, el ingeniero afronta, tal vez, otro problema. La acción basada en los principios quizá sea renunciar al empleo, aunque esta decisión bien podría afectar a otros, como a su pareja o a sus hijos.

Una situación muy difícil para los ingenieros profesionales surge cuando su empleador actúa sin ética. Es decir, una compañía es responsable del desarrollo de un sistema crítico de seguridad y, debido a presión del tiempo, falsifica los registros de validación de seguridad. ¿Es responsabilidad del ingeniero mantener la confidencialidad o alertar al cliente o manifestar, de alguna forma, que el sistema entregado quizá sea inseguro?

El problema aquí es que no hay absolutos cuando se trata de seguridad. Aunque el sistema pueda no estar validado de acuerdo con criterios predefinidos, dichos criterios quizá sean demasiado estrictos. En realidad el sistema operará con seguridad a lo largo de su vida. También está el caso de que, aun cuando se valide de manera adecuada, el sistema falle y cause un accidente. La detección oportuna de los problemas puede resultar lesiva para el empleador y otros trabajadores; el fracaso por revelar los problemas podría ser

El lector debe formar su propio criterio en estos asuntos. Aquí, la posición ética adecuada depende por completo de las percepciones de los individuos que están implicados. En este caso, el potencial de daño, el alcance del mismo y las personas afectadas deben influir en la decisión. Si el escenario es muy peligroso, estaría justificado anunciarlo a través de la prensa nacional (por ejemplo). Sin embargo, siempre hay que tratar de resolver la situación sin dejar de respetar los derechos de su empleador.

Otro conflicto ético es la participación en el desarrollo de sistemas militares y nucleares. Al respecto, algunas personas se sienten muy afectadas por estos temas y evitan participar en el desarrollo de algún sistema asociado con los sistemas militares. Otras más trabajarán en los sistemas militares, pero no en los de armamento. Incluso otras sentirán que la seguridad nacional es un principio fundamental y no tienen objeciones éticas para trabajar en sistemas de armamento.

En tal situación es importante que tanto empleadores como empleados dejen en claro con antelación sus percepciones o puntos de vista. Cuando una organización participa en trabajo militar o nuclear, debe contar con la capacidad de especificar que los empleados tienen la voluntad de aceptar cualquier trabajo asignado. De igual forma, si un empleado toma la responsabilidad y deja en claro que no quiere trabajar en tales sistemas, los empleadores no tendrán que presionarlo para que éste lo haga más tarde.

El área general de la ética y la responsabilidad profesional se vuelven más importantes conforme los sistemas intensivos en software prevalecen en cada vez más cuestiones del trabajo y la vida cotidiana. Puede considerarse desde un punto de vista filosófico, donde se tomen en cuenta los principios básicos de la ética y se analice la ética de la ingeniería de software en relación con dichos principios básicos. Éste es el enfoque que toma Laudon (1995) y, en menor medida, Huff y Martin (1995). El texto de Johnson sobre ética computacional (2001) también trata el tema desde una perspectiva filosófica.

Sin embargo, este enfoque filosófico resulta muy abstracto y difícil de relacionar con la experiencia cotidiana. Es preferible el enfoque más concreto plasmado en los códigos de conducta y práctica. Se considera que la ética se analiza mejor en un contexto de ingeniería de software y no como un tema por derecho propio. Por lo tanto, en este libro no se presentan, donde es adecuado, discusiones éticas abstractas, sino que se incluyen ejemplos en los ejercicios que son el punto de partida para una discusión grupal sobre conflictos éticos.

1.3 Estudios de caso

Para ilustrar los conceptos de la ingeniería de software, a lo largo del libro se utilizan ejemplos de tres tipos de sistemas diferentes. La razón de no usar un solo estudio de caso obedece a que uno de los mensajes clave de este libro es que la práctica de la ingeniería de software depende del tipo de sistemas a producir. Por consiguiente, se elegirá un ejemplo adecuado cuando se estudien conceptos como seguridad y confiabilidad, modelado de sistema, reutilización, etcétera.

Los tres tipos de sistemas que se usan como estudios de caso son:

1. *Un sistema embebido* Se trata de un sistema donde el software controla un dispositivo de hardware y está *embebido* en dicho dispositivo. Los conflictos en los sistemas *embebidos* incluyen por lo general tamaño físico, capacidad de reacción,