Referencia bibliográfica

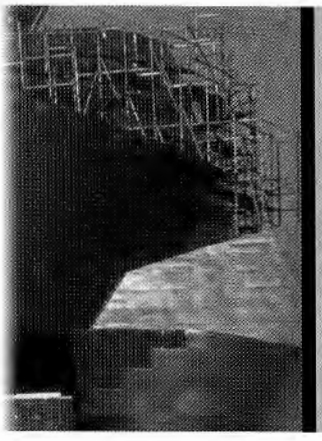
Sommerville, Ian. (2011). *Ingeniería de Software*. México: Pearson Educación. Pp. 1-9.

SOMMERVILLE

INGENIERÍA DE SOFTWARE

9

PEARSON



INGENIERÍA DE SOFTWARE

Novena edición

Ian Sommerville

Traducción:
Víctor Campos Olguín
Traductor especialista en Sistemas Computacionales

Revisión técnica:
Sergio Fuenlabrada Velázquez
Edna Martha Miranda Chávez
Miguel Ángel Torres Durán
Mario Alberto Sesma Martínez
Mario Oviedo Galdeano
José Luis López Goytia
Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas-Instituto Politécnico Nacional, México

Darío Guillermo Cardacci
Universidad Abierta Interamericana, Buenos Aires, Argentina

Marcelo Martín Marciszack
Universidad Tecnológica Nacional, Córdoba, Argentina

Addison-Wesley

México • Argentina • Brasil • Colombia • Costa Rica • Chile • Ecuador
España • Guatemala • Panamá • Perú • Puerto Rico • Uruguay • Venezuela

Datos de catalogación bibliográfica

Sommerville, Ian

Ingeniería de Software

PEARSON EDUCACIÓN, México, 2011

ISBN: 978-607-32-0603-7

Área: Computación

Formato: 18.5 × 23.5 cm Páginas: 792

Edición en español

Editor: Luis M. Cruz Castillo
e-mail: luis.cruz@pearson.com
Editor de desarrollo: Felipe Hernández Carrasco
Supervisor de producción: Juan José García Guzmán

NOVENA EDICIÓN, 2011

ISBN VERSIÓN IMPRESA: 978-607-32-0603-7
ISBN VERSIÓN E-BOOK: 978-607-32-0604-4
ISBN E-CHAPTER: 978-607-32-0605-1

PRIMERA IMPRESIÓN

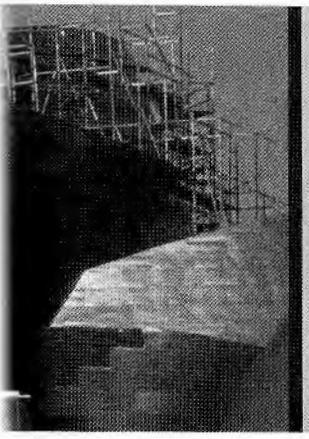
Impreso en México. *Printed in Mexico.*

1 2 3 4 5 6 7 8 9 0 - 14 13 12 11

Addison Wesley
es una marca de

PEARSON

Contenido breve



PARTE 1

Introducción a la ingeniería de software

La meta de esta parte del libro es ofrecer una introducción general a la ingeniería de software. Se incluyen conceptos importantes como procesos de software y métodos ágiles; además, se describen actividades esenciales del desarrollo de software, desde la especificación inicial del software hasta la evolución del sistema. Los capítulos de esta parte se diseñaron para apoyar un curso de un semestre en ingeniería de software.

El capítulo 1 introduce al lector de manera general en la ingeniería de software profesional y define algunos conceptos al respecto. También se escribe un breve análisis de los conflictos éticos en la ingeniería de software. Es importante que los ingenieros de software consideren las numerosas implicaciones de su trabajo. Este capítulo además presenta tres estudios de caso que se usan en el libro, a saber: un sistema para administrar registros de pacientes que se someten a tratamiento por problemas de salud mental, un sistema de control para una bomba de insulina portátil y un sistema meteorológico a campo abierto.

Los capítulos 2 y 3 tratan los procesos de ingeniería de software y el desarrollo ágil. En el capítulo 2 se presentan modelos de proceso de software genérico de uso común, como el waterfall (cascada), y se estudian las actividades básicas que son parte de dichos procesos. El capítulo 3 complementa esto con un estudio de los métodos de desarrollo ágil para ingeniería de software. Se usa sobre todo la programación extrema como

ejemplo de un método ágil, aunque también en esta sección se introduce brevemente Scrum.

Los capítulos restantes son amplias descripciones de las actividades del proceso de software que se introducirán en el capítulo 2. En el capítulo 4 se trata un tema importante de la ingeniería de requerimientos, donde se definen las necesidades de lo que debe hacer un sistema. El capítulo 5 muestra el modelado de sistemas usando el UML, donde se enfoca el uso de los diagramas de caso, diagramas de clase, diagramas de secuencia y diagramas de estado para modelar un sistema de software. El capítulo 6 introduce al diseño arquitectónico y estudia la importancia de la arquitectura y el uso de patrones arquitectónicos en el diseño de software.

El capítulo 7 trata sobre el diseño orientado a objetos y el uso de patrones de diseño. Aquí también se observan importantes problemas de implementación: reutilización, manejo de configuración y el desarrollo de host-target (anfitrión destino); también se estudia el desarrollo de código abierto. El capítulo 8 se enfoca en las pruebas del software, desde la prueba de unidad durante el desarrollo del sistema, hasta la prueba de la puesta en venta del software. También se analiza el uso del desarrollo impulsado por prueba, un enfoque pionero en los métodos ágiles, pero con gran aplicabilidad. Finalmente, el capítulo 9 brinda un panorama de los temas sobre la evolución del software. Se describen los procesos evolutivos, el mantenimiento del software y la gestión de sistemas legados.

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 de cambiar.

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:

1. *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.

2. *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 este problema.

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 mayores.

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 cálcnlo 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 frecnencia.

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 nn 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 inclnir documentación del sistema, que describe la estructura del sistema; documentación del usuario, qne 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.
¿Cuáles son las actividades fundamentales de la ingeniería de software?	Especificación, desarrollo, validación y evolución del software.
¿Cuál es la diferencia entre ingeniería de software y ciencias de la computación?	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.
¿Cuál es la diferencia entre ingeniería de software e ingeniería de sistemas?	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.
¿Cuáles son los principales retos que enfrenta la ingeniería de software?	Se enfrentan con una diversidad creciente, demandas por tiempos de distribución limitados y desarrollo de software confiable.
¿Cuáles son los costos de la ingeniería de software?	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.
¿Cuáles son los mejores métodos y técnicas de la ingeniería de software?	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.
¿Qué diferencias ha marcado la Web a la ingeniería de software?	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.

Figura 1.1 Preguntas planteadas con frecuencia sobre el 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:

1. *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

Características del producto	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.
Aceptabilidad	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.

Figura 1.2 Atributos esenciales del buen software

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.

2. *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 software.

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:

1. 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, qne 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 nna secuencia de actividades que conducen a la elaboración de un prodncto de software. Existen cnatro actividades fnndamentales que son comunes a todos los procesos de software, y éstas son:

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