Quinta edición

Ingeniería del Software

Un enfoque práctico

ROGER S. PRESSMAN

ADAPTADO POR DARREL INCE

Mc Graw Hill

CONSULTOR EDITORIAL
ÁREA DE INFORMÁTICA Y COMPUTACIÓN

Gerardo **Quiroz Vieyra**
Ingeniero de Comunicaciones y Electrónica
por la ESIME del Instituto Politécnico Nacional
Profesor de la Universidad Autónoma Metropolitana
Unidad Xochimilco
MÉXICO

INGENIERÍA DEL SOFTWARE
UN ENFOQUE PRÁCTICO
Quinta edición

Roger S. Pressman
R.S. Pressman & Associates, Inc.

ADAPTACIÓN

Darrel Ince
Open University

TRADUCCIÓN

Rafael Ojeda Martín **Isabel Morales Jareño**
Virgilio Yagüe Galaup **Salvador Sánchez Alonso**

Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software
Facultad de Informática / Escuela Universitaria de Informática
Universidad Pontificia de Salamanca *campus* Madrid (España)

Jorge A. Torres Jiménez
Director de la carrera de Ingeniería de Sistemas Computacionales
Instituto Tecnológico (TEC) de Monterrey *campus* Querétaro (México)

COLABORACIÓN

Óscar San Juan Martínez **Juana González González**
Ricardo Lozano Quesada **Lorena Esmoris Galán**

Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software
Facultad de Informática / Escuela Universitaria de Informática
Universidad Pontificia de Salamanca *campus* Madrid (España)

REVISIÓN TÉCNICA

Héctor Castán Rodríguez
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software
Facultad de Informática / Escuela Universitaria de Informática
Universidad Pontificia de Salamanca *campus* Madrid (España)

DIRECCIÓN, COORDINACIÓN Y REVISIÓN TÉCNICA

Luis Joyanes Aguilar
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software
Facultad de Informática / Escuela Universitaria de Informática
Universidad Pontificia de Salamanca *campus* Madrid (España)

MADRID • BUENOS AIRES • CARACAS • GUATEMALA • LISBOA • MÉXICO
NUEVA YORK • PANAMÁ • SAN JUAN • SANTAFÉ DE BOGOTÁ • SANTIAGO • SÃO PAULO
AUCKLAND • HAMBURGO • LONDRES • MILÁN • MONTREAL • NUEVA DELHI • PARÍS
SAN FRANCISCO • SIDNEY • SINGAPUR • ST. LOUIS • TOKIO • TORONTO

INGENIERÍA DEL SOFTWARE. Un enfoque práctico. (5.ª edición)

Traducido de la quinta edición en inglés de
SOFTWARE ENGINEERING. A Practitioner's Approach. European Adaptation

ISBN: 0-07-709677-0

ISBN: 84-481-3214-9
Depósito legal: M. 42.852-2001

Editora: Concepción Femández Madrid
Diseño de cubierta: Design Master Dima
Compuesto en FER
Impreso en: Imprenta FARESO. S. A.

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

CONTENIDOS A PRIMERA VISTA

CONTENIDO

ACERCA DEL AUTOR

ROGER S. Pressman es una autoridad internacionalmente reconocida en la mejora del proceso del Software y en las tecnologías de la Ingeniería del Software. Durante tres décadas, ha trabajado como ingeniero de Software, gestor, profesor, autor y consultor centrándose en los temas de la Ingeniería del Softwate.

Como especialista **y** gerente industrial, el Dr. Pressman ha trabajado en el desarrollo de sistemas CAD/CAM para aplicaciones avanzadas de ingeniería y fabricación. También ha ocupado puestos de responsabilidad para programación científica y de sistemas.

Después de recibir el título de Doctor en Ciencias Físicas en Ingeniería de la Universidad de Connecticut, el Dr. Pressman se dedicó a la enseñanza donde llegó a ser Profesor Asociado (Bullard Associate Professor) de Informática en la Universidad de Bridgeport **y** Director del Computer-Aided Design an Manufacturing Center en esta Universidad.

El Dr. Pressman es actualmente el Presidente de R. S. Pressman & Associates, Inc., una empresa de asesoría especializada en métodos **y** formación de Ingeniería del Softwate. Trabaja como asesor jefe, y está especializado en la asistencia a compañías para establecer unas prácticas eficientes de la Ingeniería del Software. También ha diseñado y desarrollado productos para la formación en Ingeniería del Softwate de la compañía y de mejora del proceso: *Essential Software Engineering,* un currículum en vídeo totalmente actualizado que se cuenta entre los trabajos industriales más extensos acerca de este tema **y,** *Process Advisor*, un sistema programado para la mejora en el proceso de ingeniería del softwate. Ambos productos son utilizados por cientos de compañías mundiales.

El Dr. Pressman ha escrito numerosos artículos técnicos, participa regularmente en revistas industriales, y ha publicado 6 libros. Además de *Ingeniería del Software: Un Enfoque Práctico,* ha escrito *A Manager's Guide to Software Engineering* (McGraw-Hill), un libro que hace uso de un exclusivo formato de preguntas **y** respuestas para presentar las líneas generales de Administración para implantar y comprender la tecnología; *Making Software Engineering Happen* (Prentice-Hall), que es el primer libro que trata los problemas críticos de administración asociados a la mejora del proceso del Software **y** *Software Shock* (Dorset House), un tratado centrado en el Software y su impacto en los negocios y en la sociedad. El Dr. Pressman es miembro del consejo editorial del *ZEEE Software* **y** del *Cutter IT Journal*, **y** durante muchos años fue editor de la columna «Manager» del *IEEE Software.*

El Dr. Pressman es un conocido orador, impartiendo un gran número de conferencias industriales principalmente. Ha presentado tutoriales en la Conferencia Internacional de Ingeniería del Software **y** en muchas otras conferencias industriales. Es miembro de ACM, IEEE y Tau Beta Pi, Phi Kappa Phi, Eta Kappa Nu y Pi Tau Sigma.

PREFACIO

CUANDO un software de computadora se desarrolla con éxito -cuando satisface las necesidades de las personas que lo utilizan; cuando funciona impecablemente durante mucho tiempo; cuando es fácil de modificar o incluso es más fácil de utilizar— puede cambiar todas las cosas y de hecho las cambia para mejor. Ahora bien, cuando un software de computadora falla –cuandolos usuarios no se quedan satisfechos, cuando es propenso a errores; cuando es difícil de cambiar e incluso más difícil de utilizar— pueden ocurrir y de hecho ocurren verdaderos desastres. Todos queremos desarrollar un software que haga bien las cosas, evitando que esas cosas malas merodeen por las sombras de los esfuerzos fracasados. Para tener éxito al diseñar y construir un software necesitaremos disciplina. **Es** decir, necesitaremos un enfoque de ingeniería.

Durante los 20 años que han transcurrido desde que se escribió la primera edición de este libro, la ingeniería del software ha pasado de ser una idea oscura y practicada por un grupo muy pequeño de fanáticos a ser una disciplina legítima de ingeniería. Hoy en día, esta disciplina está reconocida como un tema valioso y digno de ser investigado, de recibir un estudio concienzudo y un debate acalorado. A través de la industria, el título preferido para denominar al «programador» ha sido reemplazado por el de «ingeniero de software». Los modelos de proceso de software, los métodos de ingeniería del software y las herramientas del software se han adaptado con éxito en la enorme gama de aplicaciones de la industria.

Aunque gestores y profesionales reconocen igualmente la necesidad de un enfoque más disciplinado de software, continúan debatiendo sobre la manera en que se va a aplicar esta disciplina. Muchos individuos y compañías todavía desarrollan software de manera algo peligrosa, incluso cuando construyen sistemas para dar servicio a las tecnologías más avanzadas de hoy en día. Muchos profesionales y estudiantes no conocen los métodos nuevos **y,** como resultado, la calidad del software que se produce sufrirá y experimentará malas consecuencias. Además, se puede decir que seguirá existiendo debate y controversia a cerca de la naturaleza del enfoque de la ingeniería del software. El estado de la ingeniería es un estudio con contrastes. Las actitudes han cambiado y se ha progresado, pero todavía queda mucho por hacer antes de que la disciplina alcance una madurez total.

La quinta edición de la *Ingeniería del Software: UnEnfoque Práctico* pretende servir de guía para conseguir una disciplina de ingeniería madura. Esta edición, al igual que las cuatro anteriores, pretende servir a estudiantes y profesionales, y mantiene su encanto como guía para el profesional de industria y como una introducción completa al tema de la ingeniería para alumnos de diplomatura, o para alumnos en primer año de segundo ciclo. El formato y estilo de la quinta edición ha experimentado cambios significativos, con una presentación más agradable y cómoda para el lector, y con un contenido de acceso más fácil.

La quinta edición se puede considerar más que una simple actualización. La revisión de este libro se ha realizado para adaptarse al enorme crecimiento dentro de este campo y para enfatizar las nuevas e importantes prácticas de la ingeniería del software. Además, se ha desarrollado un sitio Web con todo lo necesario y esencial para complementar el contenido del libro. Junto con la quinta edición de *Ingeniería del Software: Un Enfoque Práctico,* la Web proporciona una gama enorme de recursos de ingeniería del software de la que se beneficiarán instructores, estudiantes y profesionales de industria.

La estructura de la quinta edición se ha establecido en cinco partes y la intención ha sido la de realizar una división por temas, y ayudar así a instructores que quizás no tengan tiempo de abarcar todo el libro en un solo trimestre. La Parte Primera, *El producto y el proceso,* es una introducción al entorno de la ingeniería del software. La intención de esta parte es introducir el tema principal y, lo que es más importante, presentar los conceptos necesarios para los capítulos siguientes. La Parte Segunda, *Gestión de proyectos de software,* presenta los temas relevantes para planificar, gestionar y controlar un proyecto de desarrollo de ingeniería. La Parte Tercera, *Métodos convencionales para la ingeniería del software,* presenta los métodos clásicos de análisis, diseño y pruebas considerados como la escuela «convencional» de la ingeniería del software. La Parte Cuarta, *Ingeniería del software orientada a objetos,* presenta los

métodos orientados a objetos a lo largo de todo el proceso de ingeniería del software, entre los que se incluyen análisis, diseño y pruebas. La Parte Quinta, *Temas avanzados en ingeniería del software*, introduce los capítulos especializados en métodos formales, en ingeniería del software de sala limpia, ingeniería del software basada en componentes, ingeniería del software cliente/servidor, ingeniería de Web, reingeniería y herramientas CASE.

La estructura de la quinta edición en cinco partes permite que el instructor «agrupe» los temas en función del tiempo disponible y de las necesidades del estudiante. Un trimestre completo se puede organizar en tomo a una de las cinco partes. Por ejemplo, un «curso de diseño» podría centrarse solo en la Parte Tercera o Cuarta; un «curso de métodos» podría presentar los capítulos seleccionados de las Partes Tercera, Cuarta y Quinta. Un «curso de gestión» haría hincapié en las Partes Primera y Segunda. Con la organización de esta quinta edición, he intentado proporcionar diferentes opciones para que el instructor pueda utilizarlo en sus clases.

El trabajo que se ha desarrollado en las cinco ediciones de *Ingeniería del Software: Un Enfoque Práctico* es el proyecto técnico de toda una vida. Los momentos de interrupción los he dedicado a recoger y organizar información de diferentes fuentes técnicas. Por esta razón, dedicaré mis agradecimientos a muchos autores de libros, trabajos y artículos, así como a la nueva generación de colaboradores en medios electrónicos (grupos de noticias, boletines informativos electrónicos y World Wide Web), quienes durante 20 años me han ido facilitando otras visiones, ideas, y comentarios. En cada uno de los capítulos aparecen referencias a todos ellos. Todos están acreditados en cuanto a **su** contribución en la rápida evolución de este campo. También quiero dedicar mis agradecimientos a los revisores de esta quinta edición. **Sus** comentarios y críticas son de un valor incalculable. Y por Último dedicar un agradecimiento y reconocimiento especiales a Bruce Maxim de la Universidad de Michigan —DearBorn, quien me ayudó a desarrollar el sitio Web que acompaña este libro, y como persona responsable de una gran parte del diseño y contenido pedadógico —.

El contenido de la quinta edición de *Ingeniería del Software: Un Enfoque Práctico* ha tomado forma gracias a profesionales de industria, profesores de universidad y estudiantes que ya habían utilizado las ediciones anteriores, **y** que han dedicado mucho tiempo en mandarme sus sugerencias, críticas e ideas. Muchas gracias también a todos vosotros. Además de mis agradecimientos personales a muchos clientes de industria de todo el mundo, quienes ciertamente me enseñan mucho **más** de lo que yo mismo puedo enseñarles.

A medida que he ido realizando todas las ediciones del libro, también han ido creciendo y madurando mis hijos Mathew y Michael. **Su** propia madurez, carácter y éxito en la vida han sido mi inspiración. Nada me ha llenado con más orgullo. Y, finalmente, a Bárbara, mi cariño y agradecimiento por animarme a elaborar esta quinta edición.

Roger S. Pressman

EL libro de Roger Pressman sobre Ingeniería del software es verdaderamente excelente: Siempre he admirado la profundidad del contenido y la habilidad del autor para describir datos difíciles de forma muy clara. De manera que tuve el honor de que McGraw-Hill me pidiera elaborar la Adaptación Europea de esta quinta edición. Esta es la tercera adaptación, y su contenido es un acopio de la quinta edición americana y el material que yo mismo he escrito para ofrecer una dimensión europea.

Las áreas del libro que contienen ese material extra son las siguientes:

- Existe una gran cantidad de material sobre métodos formales de desarrollo del software. Estas técnicas fueron pioneras en el Reino Unido y Alemania, y han llegado a formar parte de los juegos de herramientas críticos para los ingenieros de software en el desarrollo de sistemas altamente integrados y críticos para la seguridad.
- He incluido una sección sobre patrones de diseño. Estos son los que tienen lugar generalmente en miniarquitecturas que se dan una y otra vez en sistemas orientados a objetos, y que representan plantillas de diseño que se reutilizarán una y otra vez. He viajado por toda Europa, y me he encontrado constantemente compañías cuyo trabajo depende realmente de esta tecnología.
- He aportado material sobre métricas y en particular la utilización de GQM como métrica de método de desarrollo. A medida que la ingeniería del software se va integrando poco a poco dentro de una disciplina de ingeniería, esta tecnología se va convirtiendo en uno de sus fundamentos. La métrica ha sido uno de los puntos fuertes en Europa de manera que espero que toda la dedicación que he puesto en este tema lo refleje.
- He incluido también material sobre el Lenguaje de Modelado Unificado (UML) el cual refleja el gran incremento de utilización de este conjunto de notaciones en Europa Occidental. Además, sospecho que van a ser de hecho las notaciones de desarrollo de software estándar durante los próximos tres o cuatro años.
- He dado mayor énfasis al desarrollo de sistemas distribuidos mediante el lenguaje de programación Java para ilustrar la implicación de algunos de los códigos. Ahora que Internet y el comercio electrónico están en pleno auge en Europa, las compañías cada vez se vuelcan más en las técnicas de ingeniería del software al desarrollar aplicaciones distribuidas. Y esto también queda reflejado en el libro.
- He incluido también material sobre métodos de seguridad e ingeniería. Con la utilización de Intemet (una red abierta) todos los ingenieros del software tendrán que saber muchas más técnicas tales como firmas digitales y criptografía.
- Hacia el final del libro he hecho especial hincapié en la utilización de aplicaciones de comercio electrónico para mostrar de esta manera la tecnología distribuida.

Existen dos partes que hacen referencia al libro: un sitio Web americano muy importante, desarrollado por el Dr. Pressman; y un sitio de entrada, cuya dirección se proporciona a lo largo de todo el libro al final de cada capítulo. Este sitio contiene el material relevante para la edición europea y los enlaces con el sitio americano. Ambos sitios Web en combinación contienen sub-Webs con recursos para Estudiantes, para Profesores y para Profesionales.

En los *Recursos para el estudiante* se incluye una guía de estudio que resume los puntos clave del libro, una serie de autopruebas que permiten comprobar la comprensión del material, cientos de enlaces a los recursos de ingeniería del software, un caso práctico, materiales complementarios y un tablón de mensajes que permite comunicarse con otros lectores.

En la parte *Recursos para el profesor* se incluye una guía para el instructor, un conjunto de presentaciones en PowerPoint basadas en este libro, un conjunto de preguntas que se pueden utilizar para practicar mediante deberes y exámenes, un conjunto de herramientas muy pequeñas (herramientas sencillas de ingeniería del software adecuadas para su utilización en clase), una herramienta de Web que permite crear un sitio Web específico para un curso, y un tablón de mensajes que hace posible la comunicación con otros instructores.

En los *Recursos para profesionales* se incluye documentación para los procesos de ingeniería del software, listas de comprobación para actividades tales como revisiones, enlaces a proveedores de herramientas profesionales, cientos de enlaces a recursos de ingeniería del software, información sobre los paquetes de vídeo de Pressman, y comentarios y ensayos industriales acerca de varios temas de la ingeniería del software.

Ingeniería del software es un libro excelente, y espero que los aportes que he incluido para el lector europeo hagan de este libro una lectura obligada.

Darrel Ince

PRÓLOGO A LA CUARTA EDICIÓN EN ESPAÑOL

EL ESTADO DEL ARTE DE LA INGENIERÍA DEL SOFTWARE

La **Ingeniería de/l Software**[1] es una disciplina o área de la Informática o Ciencias de la Computación, que ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelven problemas de todo tipo. Hoy día es cada vez más frecuente la consideración de la ***Zngenierh de/l Software*** como una nueva área de la ingeniería, y el ***ingeniero de/l software*** comienza a ser una profesión implantada en el mundo laboral internacional, con derechos, deberes y responsabilidades que cumplir, junto a una, *ya*, reconocida consideración social en el mundo empresarial y, por suerte, para esas personas con brillante futuro.

La Ingeniería de/l Software trata con áreas muy diversas de la informática y de las ciencias de la computación, tales como construcción de compiladores, sistemas operativos o desarrollos en Intranet/Internet, abordando todas las fases del ciclo de vida del desarrollo de cualquier tipo de sistemas de información y aplicables a una infinidad de áreas tales como: negocios, investigación científica, medicina, producción, logística, banca, control de tráfico, meteorología, el mundo del derecho, la red de redes Intemet, redes Intranet y Extranet, etc.

Definición del término «Ingeniería de/l Software»

El término ***Zngenierh*** se define en el **DRAE**[2] como: «**1.** Conjunto de conocimientos y técnicas que permiten aplicar el saber científico a la utilización de la materia y de las fuentes de energía. **2.** Profesión y ejercicio del ingeniero» y el término ***ingeniero*** se define como «**1.** Persona que profesa o ejerce la ingeniería». De igual modo la Real Academia de Ciencias Exactas, Físicas y Naturales de España define el término ***Ingeniería*** como: «Conjunto de conocimientos y técnicas cuya aplicación permite la utilización racional de los materiales y de los recursos naturales, mediante invenciones, construcciones u otras realizaciones provechosas para el hombre»[3].

Evidentemente, si la Ingeniería del Software es una nueva ingeniería, parece lógico que reúna las propiedades citadas en las definiciones anteriores. Sin embargo, ni el DRAE ni la Real Academia Española de Ciencias han incluido todavía el término en sus Últimas ediciones; en consecuencia vamos a recurrir para su definición más precisa a algunos de los autores más acreditados que comenzaron en su momento a utilizar el término o bien en las definiciones dadas por organismos internacionales profesionales de prestigio tales como IEEE o ACM. Así, hemos seleccionado las siguientes definiciones de ***Zngenierh del Software:***

Definición 1

Ingeniería de Software es el estudio de los *principios* y *metodologías* para desarrollo y mantenimiento de sistemas de software [Zelkovitz, 1978][4].

Definición 2

Ingeniería del Software es la aplicación *práctica* del conocimiento científico en el diseño y construcción de programas de computadora y la *documentación* asociada requerida para desarrollar, operar (funcionar) y mantenerlos. Se conoce también como desarrollo de software o producción de software [Bohem, 19761[5].

---

[1] En Hispanoamérica. el término utilizado normalmente es: Ingeniería de Software.

[2] DRAE, Diccionario de la Real Academia Española de la Lengua.

[3] Vocabulario Científico y Técnico, edición de 1996.

[4] ZELKOVITZ, M. V., **CHAW,** A. C. y **GANNON,** J. D.: *Principles of Software Engineering* and *Design.* Prentice-Hall, Englewoods Ciif, 1979.

[5] **BOEHM,** B. W.: «Software Engineering», *IEEE Transactions* on *Computers,* C-25, núm. 12, diciembre, pp. 1226-1241.

Definición 3

Ingeniería del Software trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable que sea *fiable* y trabaje en *máquinas reales* [Bauer, 19721[6].

Definición 4

La aplicación de *un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación* (funcionamiento) y mantenimiento del software; es decir, la aplicación de ingeniería al software. 2. El estudio de enfoques como en (1) [IEEE, 19931[7].

Tomando como base la referencia de estas definiciones seleccionadas, se producen de inmediato las preguntas: *¿cuáles son las actividades que se encuadran hoy día en el mundo de la ingeniería del software?* y *¿cómo se enseña la disciplina de ingeniería del software en las universidades, escuelas de ingenieros e institutos tecnológicos?*

La respuesta a la primera pregunta se manifiesta de muy diversas formas, pero creemos que tal vez las fuentes más objetivas sean las conferencias, eventos y acontecimientos internacionales más relevantes realizados en estos Últimos años. Así, de los estudiados, hemos considerado como congresos significativos, los convocados por **SIGSOFT** (Special Interest Group on Software Engineering) de la **ACM**[8]: International Conference on Software Engineering **(ICSE,** copatrocinada con IEEE) celebrada en Boston, Massachussets, USA (17-23 de mayo de 1997) y la próxima conferencia anunciada para celebrarse en 1998 en Kyoto, Japón **(ICSE,** 19-25 de abril de 1998); 5." Symposium Foundations of Software Engineering, **SIGSOFT 97** (Zurich, 22-25 septiembre de 1997) y 6." Symposium **SIGSOFT 98** (Orlando, Florida, USA, 3-5 noviembre de 1998).

En los congresos citados anteriormente y en algunas otras fuentes como revistas de ACM/IEE y otras de tipo profesional o comercial específicas de ingeniería de software, hemos analizado sus programas, tutoriales, talleres de trabajo, contenidos, etc., y hemos seleccionado una lista con los temas más candentes del actual *estado del arte de la ingeniería del Software.* Los temas más sobresalientes son:

- Inspección de software crítico.
- Software de Tecnologías de Procesos de Negocios.
- Arquitecturas de Software Distribuido.
- Introducción a UML (Metodología de objetos, método unificado de Booch, Rumbaugh y Jacobson).
- Control técnico de proyectos software.
- Marcos de trabajo (*frameworks*) de empresa orientados a objetos.
- Una introducción a CORBA (Estándar para objetos distribuidos).
- Estrategias de ingeniería inversa para migración de software.
- Ingeniería de objetos.
- Modelado y análisis de arquitectura de software.
- Objetos distribuidos.
- Sistemas Cliente/Servidor.
- Reingeniería.
- CASE.
- Análisis y Diseño Orientados a Objetos.
- ...

Esta cuarta edición ha mejorado en cantidad y calidad a la tercera edición, pero su actualidad en el año 1997, reside en el hecho de que la mayoría de temas tratados o que se van a tratar en los congresos de la ACM (listados anteriormente), están contemplados y tratados por el autor en este libro. En cualquier forma, deseamos destacar muy expresamente algunas mejoras concretas que vienen a llenar una importante laguna que existía en los libros de ingeniería del

[6] **BAUER,** F. L.: «Software Engineering», *Information Processing,* 71, North Holland Pubiishing Co., Amsterdarn, 1972
[7] IEEE: *Standards Collection: Software Engineering,* IEEE Standard 610.12-1990, IEEE, 1993.
[8] URL de **ACM,** http://www.acm.org.

software en inglés y, por supuesto, en español. Así, destacamos el estudio y profundidad que se hace de temas tan candentes y de actualidad en el mundo de la ingeniería del software tales como: *Métodosformales, reutilización de software, reingeniería, métodos de software Cliente/Servidor,* ***CASE,*** *análisisldiseñolpruebas* y *métricas orientados a objetos, etc.,* junto con un epílogo donde muestra el estado actual y futuro de la Ingeniería del Software. Con relación a la tercera edición se aprecia una consolidación de tratamientos y una unificación de bloques de conocimiento que consiguen mejorar el aprendizaje del lector.

Una de las aportaciones más relevantes que aporta esta nueva edición es la excelente bibliografía y referencias bibliográficas que se adjuntan en cada capítulo del libro, junto a una novedad que poco a poco comienza a incorporarse en las buenas obras científicas y que aquí alcanza un excelente nivel: ***direcciones electrónicas (URLs) de sitios Web de Internet*** notables, donde se pueden ampliar conocimientos, fuentes bibliográficas, consultorías, empresas, organismos internacionales, etc., especializados en Ingeniería de Software.

En lo relativo a la segunda pregunta antes enunciada, su respuesta implica el uso de libros de texto y manuales que ayuden al estudiante a complementar las lecciones impartidas por el profesor, así como preparar sus trabajos, prácticas, exámenes, etc. Un libro para ser considerado de texto o referencia de una determinada asignatura requiere que su contenido contenga todos o la mayoría de los descriptores o tópicos considerados en el programa de la asignatura. Veamos por qué esta obra es idónea como libro de texto para asignaturas del cum'culum universitario de ***Ingeniería de/l Software.***

EL LIBRO COMO TEXTO DE REFERENCIA UNIVERSITARIA

La importancia fundamental de la disciplina *Ingeniería del Software* se está manifestando, de modo destacado, en los *currículum* de informática y ciencias de la computación de la mayoría de las universidades de todo el mundo, y seguirá creciendo su importancia a medida que nos acerquemos al tercer milenio.

Debido a estas circunstancias, las organizaciones profesionales, los departamentos educativos de los diversos gobiernos y los departamentos universitarios se han preocupado en esta década y en las anteriores de estandarizar los programas curriculares de las diferentes carreras, incluyendo materias (asignaturas) troncales y obligatorias, en los planes de estudios de Facultades y Escuelas de Ingenieros de todo el mundo.

El caso más significativo lo constituyen las organizaciones profesionales internacionales que se han preocupado también de este proceso. Entre las más destacadas sobresalen ACM (Association of Computing Machinery) e IEEE (Institute for Electrical and Electronics Engineers). Así, en el año 1991, estas dos organizaciones publicaron conjuntamente unas recomendaciones con los requisitos (materias) imprescindibles que, al menos, debían contemplar todos los planes de estudios de carreras relacionadas con Ciencias de la Computación (Informática). Estas recomendaciones han sido seguidas por todas las universidades de Estados Unidos y prácticamente, de una u otra forma, por casi todas las universidades europeas e hispanoamericanas, desde el año de su publicación.

Las recomendaciones ACM/IEEE[9] dividen los requisitos del currículum en nueve áreas diferentes, con subdivisiones en esas áreas. Para cada subárea se recomienda un número mínimo de unidades de conocimiento *(knowledge units)* con una indicación de tiempo para cada una de ellas (períodos de 50 minutos/clase). En estas nueve áreas se incluyen: Algoritmos, Arquitectura, Inteligencia Artificial, Bases de Datos, Interfaces Hombre/Máquina, Computación numérica, Sistemas Operativos, Programación, Ingeniería del Software, Lenguajes de Programación y Temas legales, profesionales y sociales. Los temas recomendados en el área de Ingeniería de Software son:

1. Conceptos fundamentales de resolución de problemas.
2. Proceso de desarrollo de software.
3. Especificaciones y requisitos de software.
4. Diseño e implementación de software.
5. Verificación y validación.

[9] *http:/www acm.org ; http:/xavier.xu.edu:8000/~~lewan/new cum'culum.html*

En España, el Consejo de Universidades, organismo encargado de dictar directrices y normas para los planes de estudios de las universidades tiene redactadas las normativas que han de cumplir todas las universidades que deseen impartir las carreras de Ingeniería en Informática (cinco años académicos, diez semestres), Ingeniería Técnica en Informática de Sistemas e Ingeniería Técnica en Informática de Gestión (tres años académicos, seis semestres) y se publican oficialmente en el *Boletín Oficial del Estado (BOE)*.En estas normativas de obligado cumplimiento por todas las universidades, se incluyen las materias (asignaturas o conjunto de asignaturas) troncales que se deben incluir obligatoriamente en todos los planes de estudios, además de otras asignaturas con carácter obligatorio y opcional que cada universidad fija en sus planes de estudios.

Ingeniería del Software es una materia troncal incluida en las carreras citadas. 18 créditos (cada crédito equivale a diez horas de clase) en la ingeniería superior *(ingeniero informútico)* y 12 créditos en la ingeniería técnica de informática de gestión *(ingeniero técnico informútico)*. Esto significa una carga lectiva considerable que todos los estudiantes de carreras de informática y de computación han de estudiar, normalmente en los cursos 3.º a 5.º

Este libro puede ser utilizado en diferentes tipos de cursos de ingeniería de software tanto a nivel universitario como a nivel profesional:

1. Cursos introductorios de Ingeniería del Software. Estudiantes de primer ciclo (tres años) y segundo ciclo (cinco años), o en carreras de cuatro años (como suele ser el caso de las universidades hispanoamericanas y alguna española), sin experiencia previa en Ingeniería del Software, pero con formación de dos o tres cursos universitarios (cuatro o cinco semestres) al menos.
2. Cursos introductorios y de nivel medio sobre temas específicos de Ingeniería del Software tales como análisis de requisitos y especificaciones, gestión de proyectos de software, métodos convencionales de Ingeniería del Software, etc.
3. Cursos avanzados en temas de Ingeniería del Software tales como: análisis y diseño orientados a objetos, métricas, ingeniería inversa, Cliente/Servidor, Reutilización de software, etcétera.

Este libro explica todos los temas incluidos en la asignatura o materia troncal ***Ingeniería del Software*** por el Consejo de Universidades de España, así como las unidades SE2 a SE5 (la unidad SE1 se refiere a conceptos de tipo general que suelen incluirse en otras asignaturas básicas) del currículum de la ACM/IEEE de 1991. Por nuestro conocimiento personal (conferencias, cursillos, estancias...) de muchas universidades hispanoamericanas, nos consta que los planes de estudio de estas universidades incluyen también asignaturas de tipo obligatorio de Ingeniería del Software que siguen prácticamente las recomendaciones de ACM/IEEE y son muy similares a los programas que se siguen también en España.

APÉNDICE

La obra actual incluye gran cantidad de siglas y acrónimos en inglés, la mayoría de las cuales, exceptuando las ya acreditadas en inglés cómo BPR..., se han traducido al español. Por su especial importancia y la gran cantidad de ellas incluidas en el libro, el equipo de traducción decidimos recopilar todas las siglas en inglés y sus traducciones al español; a continuación se ha construido dos tablas ordenadas alfabéticamente en inglés y en español, con el objetivo principal de que el lector pueda localizar fácilmente cualquiera de las siglas que aparecen en el texto, y en su caso, la traducción al español. Estas tablas se presentaron a la editora de la obra que tras ver el servicio que proporcionaba al lector, aceptó incluirlas como Apéndice. De este modo, el lector podrá comprobar en cualquier momento entradas de siglas tanto en español como en inglés.

EPÍLOGO

La traducción de esta obra ha sido un esfuerzo común que hemos realizado profesores de diferentes universidades españolas e hispanoamericanas —profesores de Ingeniería del Software que hemos utilizado, en la mayoría de los casos, las ediciones anteriores de esta obra como libro de referencia en nuestras clases y continuaremos utilizándolo. Por esta circunstancia, hemos

podido apreciar que esta cuarta edición ha mejorado de modo muy notable a las ediciones anteriores y tenemos el convencimiento de que los principios y conceptos considerados en ella, seguirán teniendo una influencia considerable en numerosos estudiantes de ingeniería, licenciatura informática o sistemas computacionales de habla española, como nos consta que siguen influyendo en el mundo de habla inglesa. Estamos seguros de que serán muchos los estudiantes que seguirán formándose en Ingeniería del Software con este libro como referencia fundamental.

Esta cuarta edición llena numerosas lagunas en la literatura de habla española, ya que actualiza los contenidos tradicionales de la Ingeniería del Software y los extiende a los temas avanzados modernos que ya hemos considerado. El excelente trabajo de Roger S. Pressman permitirá seguir formando numerosos y buenos profesionales de Ingeniería del Software para que esta industria pueda seguir desarrollándose.

Madrid, septiembre de 1997

Luis Joyanes Aguilar
Director del Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software
Facultad de Informática y Escuela Universitaria de Informática
Universidad Pontificia de Salamanca. Campus de Madrid

PRÓLOGO A LA QUINTA EDICIÓN EN ESPAÑOL

EL siglo XXI se enfrenta a la creciente implantación de la sociedad del conocimiento. La era del conocimiento en que vivimos no sólo está cambiando la sociedad en sí misma, sino que los nuevos modelos de negocios requieren la reformulación de nuevos conceptos. Conocimiento, activos intangibles, Web, etc., son algunos de los términos más utilizados en cualquier ambiente o negociación. Esta era del conocimiento requiere de nuevas tendencias apoyadas precisamente en el conocimiento. La ingeniería del software no es una excepción, y por ello se requiere no sólo una actualización de conceptos, sino también una comprensión y una formulación del nuevo conocimiento existente en torno a las nuevas innovaciones y teorías de dicha disciplina.

En cada edición de su clásica obra, Roger Pressman nos tiene acostumbrados a la sorpresa y a la admiración por la clara y excelente exposición de los temas tratados. Esta vez tampoco ha sido una excepción, muy al contrario, Pressman da la sensación de haber conseguido «la cuadratura del círculo» o mejor aún, ha encontrado la piedra filosofal para formar y educar a los actuales y —sobre todo— futuros ingenieros de software del futuro (o a los ingenieros informáticos e ingenieros de sistemas y licenciados en informática que se forman en esta disciplina). En esta quinta edición, Pressman proporciona al lector una ingente cantidad de conocimiento relativo a ingeniería del software que facilitará considerablemente su formación con todo el rigor profesional y científico que requiere la nueva era del conocimiento que viviremos en esta década.

EL NUEVO CONTENIDO

Una de las grandes y atractivas novedades de esta quinta edición es su nuevo formato y estilo. El SEPA 5/2, como se le conoce en la versión en inglés, ha mejorado el libro y lo ha hecho más legible y atractivo al lector. Mediante iconos y una lista normalizada de seis cuestiones clave, Pressman va guiando al lector sobre los temas mas importantes de cada capítulo a la vez que su lectura le introduce paulatina e inteligentemente en las ideas y conceptos más importantes. Esta quinta edición contiene todos los temas importantes de la cuarta edición e introduce una gran cantidad de temas relativos a las nuevas tendencias, herramientas y metodologías que plantea la ingeniería de software actual y futura, así como la naciente nueva ingeniería Web. Un estudio detenido del contenido nos conduce a los cambios más sobresalientes realizados en esta quinta edición, que son, entre otros, los siguientes:

- Cinco nuevos capítulos (Capítulo 14, Diseño arquitectónico; Capítulo 15, Diseño de la interfaz de usuario, proporcionando reglas de diseño, procesos de modelado de interfaces, diseño de tareas y métodos de evaluación; Capítulo 16, Diseño a nivel de componentes; Capítulo 27, examina los procesos y la tecnología de la ingeniería de software basada en componentes, y, Capítulo 29, que presenta los nuevos conceptos de Ingeniería Web (procesos WebE, análisis y formulación de aplicaciones Web, es decir arquitectura, navegación y diseño de interfaces, pruebas y aplicaciones Web y gestión de proyectos de ingeniería Web).
- Gran cantidad de actualizaciones y revisiones de los 32 capítulos de su contenido total. Los cambios clave son numerosos, y los más sobresalientes son:

  —Modelos de procesos evolutivos (WinWin) y de ingeniería basada en componentes.
  —Nuevas secciones sobre control estadístico de la calidad.
  —Modelo de estimación de COCOMO 11.
  —Técnicas a prueba de errores para gestión de calidad de software **(SQA).**
  —Ingeniería de requisitos.
  —El lenguaje unificado de modelado, UML (Unified Modeling Language).
  —Nuevas regias y teoría de calidad de software que siguen la normativa ISO 9000.

NUEVOS RECURSOS DOCENTES Y PROFESIONALES

Si la edición en papel que tiene el lector en sus manos ya es de por sí excelente, el sitio Web del libro no le queda a la zaga (*www.pressman5.com*). Este sitio es una de las mejores herramientas de las que pueden disponer estudiantes, profesores y maestros, y profesionales del mundo del software. Destaquemos algunas.

Recursos de estudiantes

- Guía de estudios.
- Autotest.
- Recursos basados en la Web.
- Estudio de casos.
- Vídeos.
- Contenidos suplementarios.
- Foros para intercambios de mensajes entre lectores.

Recursos de profesores y maestros

- Guía del profesor.
- Transparencias (acetatos) en PowerPoint.
- Bancos de prueba.
- Herramientas de ingeniería de software.
- Foros de intercambio de mensajes entre colegas.

Recursos profesionales

- Plantillas de productos de documentos y de trabajos.
- Listas de pruebas de ingeniería de software.
- Herramientas de ingeniería de software.
- Herramientas CASE.
- Recursos de ingeniería de software.
- Modelos de procesos adaptables.
- Currículum de vídeos de calidad para la industria.
- Comentarios de la industria.
- Foros profesionales.

EL GLOSARIO DE SIGLAS: UN NUEVO APÉNDICE

Una de las características más sobresalientes de esta obra es que recoge con gran profusión la ingente cantidad de siglas que hoy día se utilizan en la industria del software junto a otras muchas más acuñadas por el propio autor.

El equipo de profesores que ha traducido y adaptado la versión en inglés de común acuerdo con la editora acordó realizar un apéndice que incluyera todas las siglas incluidas en el libro y las traducciones correspondientes en español, y viceversa. Este apéndice busca, al igual que ya se hiciera en la segunda edición en español, facilitar al lector la lectura y seguimiento del modo más fácil posible y que le permita hacer la correspondencia entre ambos idiomas cuado lo necesite. Por ello, este apéndice contiene un diccionario inglés-español y otro español-inglés de siglas. El método que se ha seguido durante la traducción ha sido traducir prácticamente todas las siglas, y sólo se han realizado algunas excepciones, como **SQA** (Software Quality Assurement) por su uso frecuente en la jerga informática, aunque en este caso hemos utilizado ambos términos (en inglés, **SQA** y en español, **GCS,** Gestión de Calidad del Software). En este caso, estas siglas en español coinciden con Gestión de Configuración del Software **(GCS),** por lo que a veces estas siglas se han traducido por GCVS (Gestión de Configuración de Versiones de Soft-

ware) para evitar duplicidades. Ésta es una de las razones fundamentales por la que hemos incluido el glosario de siglas.

EL EQUIPO TRADUCTOR

La edición británica ha sido adaptada por un prestigioso profesor británico, y la edición y la adaptación españolas han sido realizadas por un numeroso equipo de profesores del *Departamento de Lenguajes y Sistemas Informáticos e ingeniería del Software* de la *Facultad de informática y Escuela Universitaria de Informática* de la **Universidad Pontificia de Salamanca (España)** del *campus* de Madrid, que ha contado con la inestimable colaboración de profesores del prestigioso **Instituto Tecnológico (TEC) de Monterrey,** de su *campus* de Querétaro (México). Una obra de la envergadura de esta quinta edición requería —como ya sucedió tambien en la edición anterior— del trabajo y coordinación de numerosas personas. Muchas han sido las horas dedicadas a la traducción, adaptación y sucesivas revisones de galeradas de las pruebas de imprenta. Confiamos haber cumplido nuestra obligación con dignidad y profesionalidad.

EL FUTURO ANUNCIADO

Esta quinta edición ha sido actualizada sustancialmente con nuevos materiales que reflejan el estado del arte de la ingeniería del software. El material obsoleto se ha eliminado de esta edición para crear un texto fácil de leer y entender, **y** totalmente actualizado. Sin embargo, y con ser importante todo su vasto y excelente contenido, queremos destacar que esta nueva edición nos brinda y abre el camino al futuro que señala la moderna ingeniería de software basada en objetos y componentes, así como a la ingeniería Web, conservando el rigor y la madurez de las teorías ya clásicas de la ingeniería del software.

Sin lugar a dudas, Pressman ha conseguido una excelente obra, y en una prueba clara de profesionalidad y excelencia ha logrado superar sus cuatro ediciones anteriores ya de por **sí** excelentes.

Madrid y Carchelejo (España), verano de 2001

Luis Joyanes Aguilar
Director del Departamento de Lenguajes, Sistemas informáticos e Ingeniería de Software
Facultad de Informática/Escuela Universitaria de Informática
Universidad Pontificia de Salamanca *campus* Madrid

UTILIZACIÓN DEL LIBRO

LA quinta edición de *Ingeniería del Software: Un Enfoque Práctico* se ha vuelto a diseñar para aumentar la experiencia del lector y para proporcionar enlaces integrados al sitio Web, **http://www.pressman5.com.** Dentro de este sitio los lectores encontrarán información complementaria útil y rica, y una gran cantidad de recursos para los instructores que hayan optado por este libro de texto en clase.

A lo largo de todo el libro se van encontrando iconos que deberán interpretarse de la siguiente manera:

Utilizado para enfarizar un punto importante en el cuerpo del texto.

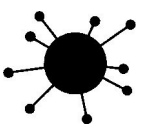
El icono de **punto clave** ayudará a encontrar de forma rápida los puntos importantes.

Para punteros que conducirán directamente a los recursos de la Web.

El icono **Referencia web** proporciona punteros directos a sitios Web importantes relacionados con la ingeniería del software.

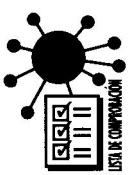
Un consejo práctico de/mundo red aplicado a la ingeniería del software.

El icono de **consejo** proporciona una guía pragmática que servirá de ayuda para tomar la decisión adecuada o evitar los problemas típicos al elaborar software.


Un tema seleccionado.

El icono de **sitio Web** indica que se dispone de más información sobre el tema destacado en el sitio Web SEPA.

El icono de **signo de interrogación** formula preguntas que se responderán en el cuerpo del texto.



lista de comprobación

El icono de **lista de comprobación** nos señala las listas de comprobación que ayudan a evaluar el trabajo de ingeniería del software que se está llevando a cabo y los productos del trabajo.

Proporciona una referencia cruzada importante dentro del libro

El icono de **referencia cruzada** nos conducirá a alguna otra parte del texto en donde se podrá encontrar información relevante sobre el tema en cuestión.

Documento

El icono de **documento** nos señala los bocetos, descripciones y ejemplos de documentos del sitio Web SEPA.

El icono de **cita** presenta citas interesantes que tienen gran relevancia para los temas que se estén tratando.

PARTE

I

EL PRODUCTO Y EL PROCESO

En esta parte de *Ingeniería del software: un enfoque práctico* aprenderá sobre el producto que va a ser tratado con ingeniería y el proceso que proporciona un marco de trabajo para la tecnología de Ingeniería del software. En los capítulos siguientes se tratan las preguntas:

- ¿Qué es realmente el software de computadora?
- ¿Por qué se lucha para construir sistemas de alta calidad basados en computadoras?
- ¿Cómo se pueden establecer categorías de dominios de aplicaciones para software de computadoras?
- ¿Qué mitos de software van a existir?
- ¿Qué es un «proceso» de software?
- ¿Existe una forma genérica de evaluar la calidad de un proceso?
- ¿Qué modelos de procesos se pueden aplicar al desarrollo del software?
- ¿En que difieren los modelos de proceso lineales e iterativos?
- ¿Cuáles son sus puntos fuertes y débiles?
- ¿Qué modelos de proceso avanzados se han propuesto para la ingeniería del software?

Una vez contestadas todas estas preguntas, estará más preparado para comprender los aspectos técnicos y de gestión de la disciplina de ingeniería a la que se dedica el resto del libro.

CAPÍTULO

1 EL PRODUCTO

LAS alarmas comenzaron más de una década antes del acontecimiento. Con menos de dos años a la fecha señalada, los medios de comunicación recogieron la historia. Los oficiales del gobierno expresaron su preocupación, los directores de la industria y de los negocios comprometieron grandes cantidades de dinero, y por Último, las advertencias horribles de catástrofe llegaron a la conciencia del público. El software, al igual que el ahora famoso error Y2K, podría fallar, y como resultado, detener el mundo como nosotros lo conocimos.

Como vimos durante los últimos meses del año 1999, sin querer, no puedo dejar de pensar en el párrafo profético contenido en la primera página de la cuarta edición de este libro. Decía:

> El software de computadora se ha convertido en el ***alma** mater*. Es la máquina que conduce a la toma de decisiones comerciales. Sirve de base para la investigación científica moderna y de resolución de problemas de ingeniería. Es el factor clave que diferencia los productos y servicios modernos. Está inmerso en sistemas de todo tipo: de transportes, médicos, de telecomunicaciones, militares, procesos industriales, entretenimientos, productos de oficina..., la lista es casi interminable. El software **es** casi ineludible en un mundo moderno. **A** medida que nos adentremos en el siglo XXI, será el que nos conduzca a nuevos avances en todo, desde la educación elemental a la ingeniería genética.

VISTAZO RÁPIDO

¿Qué es? El software de computadora es el producto que diseñan y construyen los ingenieros del software. Esto abarca programas que se ejecutan dentro de una computadora de cualquier tamaño y arquitectura, documentos que comprenden formularios virtuales e impresos y datos que combinan números y texto y también incluyen representaciones de información de audio, vídeo e imágenes.

¿Quién lo hace? Los ingenieros de software lo construyen, y virtualmente cualquier persona en el mundo industrializado lo utiliza bien directa o indirectamente.

¿Por qué es importante? Porque afecta muy de cerca a cualquier aspecto de nuestra vida y está muy extendido en nuestro comercio, cultura y en nuestras actividades cotidianas.

¿Cuáles son los pasos? Construir software de computadora como construimos cualquier otro producto satisfactorio, aplicando un proceso que conduce a un resultado de alta calidad que satisface las necesidades de la gente que usará el producto. Debes aplicar un enfoque de ingeniería de software.

¿Cuál es el producto obtenido? Desde el punto de vista de un ingeniero de software, el producto obtenido son los programas, documentos y los datos que configuran el software de computadora. Pero desde el punto de vista de los usuarios el producto obtenido es la información resultante que hace de algún modo el mundo mejor a los usuarios.

¿Cómo puedo estar seguro de que lo he hecho correctamente? Lee el resto de este libro, selecciona aquellas ideas que son aplicables al software que construyes y aplícalas a tu trabajo.

Cinco años después de que la cuarta edición de este libro fue escrita, el papel del software como «alma mater» ha llegado a ser más obvio. Un director de software de Intemet ha producido su propia economía de 500 billones de Euros. En la euforia creada por la promesa de un paradigma económico nuevo, los inversores de Wall Street dieron a las pequeñas empresas «punto-com» estimaciones en billones de dólares antes de que éstas comenzasen a producir un dólar en ventas. Han surgido nuevas industrias dirigidas por software y las antiguas que no se han adaptado a esta nueva tendencia están ahora amenazadas de extinción. El gobierno de Estados Unidos ha mantenido un contencioso frente a la mayor compañía de la industria del software, como lo mantuvo hace poco tiempo cuando se movilizó para detener las actividades monopolísticas en las industrias del acero y del aceite.

El impacto del software en nuestra sociedad y en la cultura continúa siendo profundo. Al mismo tiempo que crece su importancia, la comunidad del software trata continuamente de desarrollar tecnologías que hagan más sencillo, rápido y menos costosa la construcción de programas de computadora de alta calidad.

Este libro presenta un marco de trabajo que puede ser usado por aquellos que construyen software informático —aquellos que lo deben hacer bien—. La tecnología que comprende un proceso, un juego de métodos y un conjunto de herramientas **se** llama *ingeniería **del** software.*

1.1. LA EVOLUCIÓN DEL SOFTWARE

Hoy en día el software tiene un doble papel. Es un producto y, al mismo tiempo, el vehículo para entregarlo. Como producto, hace entrega de la potencia informática que incorpora el hardware informático o, más ampliamente, una red de computadoras que es accesible por hardware local. Si reside dentro de un teléfono celular u opera dentro de una computadora central, el software es un transformador de información, produciendo, gestionando, adquiriendo, modificando, mostrando o transmitiendo información que puede ser tan simple como un solo bit, o tan complejo como una presentación en multimedia. Como vehículo utilizado para hacer entrega del producto, el software actúa como la base de control de la computadora (sistemas operativos), la comunicación de información (redes) y la creación y control de otros programas (herramientas de software y entomos).

PUNTO CLAVE

El software es tonto un producto, como el vehículo poro su entrego

El papel del software informático ha sufrido un cambio significativo durante un periodo de tiempo superior a 50 años. Enormes mejoras en rendimiento del hardware, profundos cambios de arquitecturas informáticas, grandes aumentos de memoria y capacidad de almacenamiento y una gran variedad de opciones de entrada y salida han conducido a sistemas más sofisticados y más complejos basados en computadora. La sofisticación y la complejidad pueden producir resultados deslumbrantes cuando un sistema tiene éxito, pero también pueden suponer grandes problemas para aquellos que deben construir sistemas complejos.

Libros populares publicados durante los años 70 **y** 80 proporcionan una visión histórica útil dentro de la percepción cambiante de las computadoras y del software, **y** de **su** impacto en nuestra cultura. Osborne [OSB79] hablaba de una «nueva revolución industrial». Toffler [TOF80] llamó a la llegada de componentes microelectrónicos la «tercera ola del cambio» en la historia de la humanidad, **y** Naisbitt [NAI82] predijo la transformación de la sociedad industrial a una «sociedad de información». Feigenbaum y McCorduck [FEI83] sugirieron que la información **y** el conocimiento (controlados por computadora) serían el foco de poder del siglo veintiuno, y Stoll [STO89] argumentó que la «comunidad electrónica» creada mediante redes **y** software es la clave para el intercambio de conocimiento alrededor del mundo.

Al comienzo de los años 90, Toffler [TOF90] describió un «cambio de poder» en el que las viejas estructuras de poder (gubernamentales, educativas, industriales, económicas **y** militares) se desintegrarían a medida que

Cita:

Me introduje en el futuro, más allá de lo que el ojo humano puede ver. Tuve una visión del mundo y todo lo maravilloso que podría ser.
Tennyson

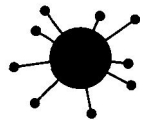
las computadoras y el software nos llevaran a la «democratización del conocimiento». A Yourdon [YOU92] le preocupaba que las compañías en Estados Unidos pudieran perder **su** competitividad en empresas relativas al software y predijo «el declive y la caída del programador americano». Hammer y Champy [HAM93] argumentaron que las tecnologías de información iban a desempeñar el papel principal en la «reingeniería de la compañía». A mediados de los años 90, la persistencia de las computadoras y del software generó una erupción de libros por «neo-Luddites» (por ejemplo: *Resisting the Virtual Life,* editado por James Brook y Ian Boal, y *The Future Does not Compute* de Stephen Talbot). Estos autores critican enormemente la computadora, haciendo énfasis en preocupaciones legítimas pero ignorando los profundos beneficios que se han llevado a cabo [LEV95].

Cita:

Las computadoras hacen las cosas más fáciles, pero la mayoría de las cosas que facilitan no es preciso hacerlas.
Andy Rooney

Al final de los años 90, Yourdon [YOU96] volvió a evaluar las perspectivas del software profesional y sugirió la «resurrección y elevación» del programador americano. A medida que internet creció en importancia, **su** cambio de pensamiento demostró ser correcto. Al final del siglo veinte, el enfoque cambió una vez más. Aquí tuvo lugar el impacto de la «bomba de relojería» Y2K (por ejemplo: [YOU98b], [DEJ98], [KAR99]). Aunque muchos vieron las predicciones de los críticos del Y2K como reacciones, sus populares lecturas devolvieron la difusión del software a sus vidas. Hoy en día, «la computación omnipresente» [NOR98] ha producido una generación de aplicaciones de información que tienen conexión en banda ancha a la Web para proporcionar «una capa de conexión sobre nuestras casas, oficinas, y autopistas» [LEV99]. El papel del software continúa su expansión.

El programador solitario de antaño ha sido reemplazado por un equipo de especialistas del software, cada uno centrado en una parte de la tecnología requerida para entregar una aplicación concreta. Y de este modo, las cuestiones que se preguntaba el programador solitario son las mismas cuestiones que nos preguntamos cuando construimos sistemas modernos basados en computadoras:



Estadísticas globales de software

- ¿Por qué lleva tanto tiempo terminar los programas?
- ¿Por qué son tan elevados los costes de desarrollo?
- ¿Por qué no podemos encontrar todos los errores antes de entregar el software a nuestros clientes?
- ¿Por qué nos resulta difícil constatar el progreso conforme se desarrolla el software?

1.2 EL SOFTWARE

En 1970, menos del uno por ciento de las personas podría haber descrito inteligentemente lo que significaba «software de computadora». Hoy, la mayoría de los profesionales y muchas personas en general piensan en su mayoría que comprenden el software. ¿Pero lo entienden realmente?

1.2.1. Características del software

Para poder comprender lo que es el software (y consecuentemente la ingeniería del software), es importante examinar las características del software que lo diferencian de otras cosas que los hombres pueden construir. Cuando se construye hardware, el proceso creativo humano (análisis, diseño, construcción, prueba) se traduce finalmente en una forma física. Si construimos una nueva computadora, nuestro boceto inicial, diagramas formales de diseño y prototipo de prueba, evolucionan hacia un producto físico (chips, tarjetas de circuitos impresos, fuentes de potencia, etc.).

El software es un elemento del sistema que es lógico, en lugar de físico. Por tanto el software tiene unas características considerablemente distintas a las del hardware:

PUNTO CLAVE

El software se desarrolla, no se fabrica.

1. El software se desarrolla, no se fabrica en un sentido clásico.

Aunque existen similitudes entre el desarrollo del software y la construcción del hardware, ambas actividades son fundamentalmente diferentes. En ambas actividades la buena calidad se adquiere mediante un buen diseño, pero la fase de construcción del hardware puede introducir problemas de calidad que no existen (o son fácilmente corregibles) en el software. Ambas actividades dependen de las personas, pero la relación entre las personas dedicadas y el trabajo realizado es completamente diferente para el software (véase el Capítulo 7). Ambas actividades requieren la construcción de un «producto» pero los enfoques son diferentes.

Los costes del software se encuentran en la ingeniería. Esto significa que los proyectos de software no se pueden gestionar como si fueran proyectos de fabricación.

2. ***El software no se «estropea».***

La Figura **1.1** describe, para el hardware, la proporción de fallos como una función del tiempo. Esa relación, denominada frecuentemente «curva de bañera», indica que el hardware exhibe relativamente muchos fallos al principio de su vida (estos fallos son atribuibles normalmente a defectos del diseño o de la fabricación); una vez corregidos los defectos, la tasa de fallos cae hasta un nivel estacionario (bastante bajo, con un poco de optimismo) donde permanece durante un cierto periodo de tiempo. Sin embargo, conforme pasa el tiempo, el hardware empieza a desgastarse y la tasa de fallos se incrementa.

El software no es susceptible a los males del entorno que hacen que el hardware se estropee. Por tanto, en teoría, la curva de fallos para el software tendría la forma que muestra la Figura **1.2.** Los defectos no detectados haran que falle el programa durante las primeras etapas de su vida. Sin embargo, una vez que se corrigen (suponiendo que no se introducen nuevos errores) la curva se aplana, como se muestra. La curva idealizada es una gran simplificación de los modelos reales de fallos del software (vease más información en el Capítulo **8).** Sin embargo la implicación es clara, el software no se estropea. ¡Pero se deteriora!

PUNTO CLAVE

El software no se estropea, pero se deteriora.

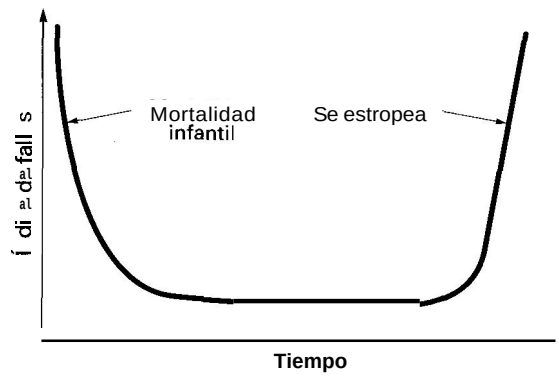


FIGURA 1.1. Curva de fallos del hardware.

Esto que parece una contradicción, puede comprenderse mejor considerando «la curva actual» mostrada en la Figura 1.2. Durante su vida, el software sufre cambios (mantenimiento). Conforme se hacen los cambios, es bastante probable que se introduzcan nuevos defectos, haciendo que la curva de fallos tenga picos como se ve en la Figura 1.2. Antes de que la curva pueda volver al estado estacionario original, se solicita otro cambio, haciendo que de nuevo se cree otro pico. Lentamente, el nivel mínimo de fallos comienza a crecer —el software se va deteriorando debido a los cambios—.

Otro aspecto de ese deterioro ilustra la diferencia entre el hardware y el software. Cuando un componente de hardware se estropea se sustituye por una pieza de repuesto. No hay piezas de repuesto para el software. Cada fallo en el software indica un error en el diseño o en el proceso mediante el que se tradujo el diseño a código máquina ejecutable. Por tanto, el mantenimiento del software tiene una complejidad considerablemente mayor que la del mantenimiento del hardware.

3. *Aunque la industria tiende a ensamblar componentes, la mayoría del software se construye a medida.*

Consideremos la forma en la que se diseña y se construye el hardware de control para un producto basado en computadora. El ingeniero de diseño construye un sencillo esquema de la circuitería digital, hace algún análisis fundamental para asegurar que se consigue la función adecuada y va al armario donde se encuentran los catálogos de componentes digitales. Después de seleccionar cada componente, puede solicitarse la compra.

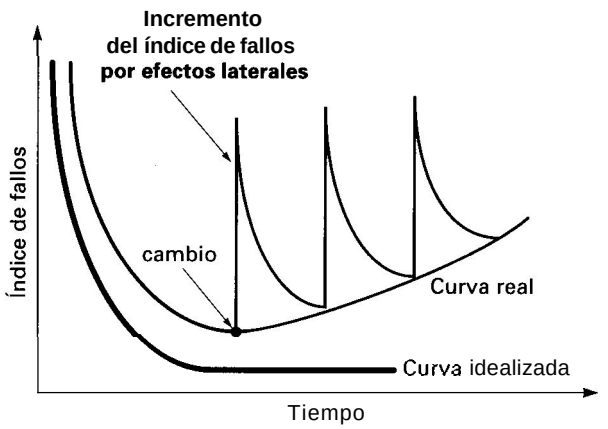


FIGURA 1.2. Curvas de fallos real e idealizada del software.

PUNTO CLAVE

Los métodos de ingeniería de software se esfuerzan para reducir la magnitud de los picos y la inclinación de la curva (Fig. 1.2).

A medida que la disciplina del software evoluciona, se crea un grupo de componentes de diseño estándar. Tornillos estándar y circuitos integrados preparados para la venta son solamente los dos mil coinponentes estándar que utilizan ingenieros mecánicos y eléctricos cuando diseñan nuevos sistemas. Los componentes reutilizables se han creado para que el ingeniero pueda concentrarse en elementos verdaderamente innovadores de un diseño, por ejemplo, las partes del diseño que representan algo nuevo. En el mundo del hardware, la reutilización de componentes es una parte natural del proceso de ingeniería. En el mundo del software es algo que sólo ha comenzado a lograrse en una escala amplia.

El componente de software debería diseñarse e implementarse para que pueda volver a ser reutilizado en muchos programas diferentes. En los años 60, se construyeron bibliotecas de subrutinas científicas reutilizables en una amplia serie de aplicaciones científicas y de ingeniería. Esas bibliotecas de subrutinas reutilizaban de forma efectiva algoritmos bien definidos, pero tenían un dominio de aplicación limitado. Hoy en día, hemos extendido nuestra visión de reutilización para abarcar no sólo los algorítmos, sino también estructuras de datos. Los componentes reutilizables modernos encapsulan tanto datos como procesos que se aplican a los datos, permitiendo al ingeniero del software crear nuevas aplicaciones a partir de las partes reutilizables. Por ejemplo, las interfaces gráficas de usuario de hoy en día se construyen frecuentemente a partir de componentes reutilizables que permiten la creación de ventanas gráficas, de menús despleglables y de una amplia variedad de mecanismos de interacción.

C VE

La mayoría del software sigue construyéndose a medida.

1.2.2. Aplicaciones del software

El software puede aplicarse en cualquier situación en la que se haya definido previamente un conjunto específico de pasos procedimentales (es decir, un algoritmo) (excepciones notables a esta regla son el software de los sistemas expertos y de redes neuronales). El contenido y el determinismo de la información son factores importantes a considerar para determinar la naturaleza de una aplicación de software. El contenido se refiere al significado y a la forma de la información de entrada y salida. Por ejemplo, muchas aplicaciones bancarias usan unos datos de entrada muy estructurados (una base de datos) y producen «informes» con determinados formatos. El software que controla una máquina automática (por ejemplo: un control numérico) acepta elementos de datos discretos con una estructura limitada y produce órdenes concretas para la máquina en rápida sucesión.

Referencia cruzada

La revolución del software se foto en el Capítulo 13. La ingeniería de software basada en componentes se presento en el Capítulo 27.

El determinismo de la información se refiere a la predecibilidad del orden y del tiempo de llegada de los datos. Un programa de análisis de ingeniería acepta datos que están en un orden predefinido, ejecuta el algoritmo(s) de análisis sin interrupción y produce los datos resultantes en un informe o formato gráfico. Se dice que tales aplicaciones son determinadas. Un sistema operativo multiusuario, por otra parte, acepta entradas que tienen un contenido variado y que se producen en instantes arbitrarios, ejecuta algoritmos que pueden ser interrumpidos por condiciones externas y produce una salida que depende de una función del entorno y del tiempo. Las aplicaciones con estas características se dice que son indeterminadas.

Algunas veces es difícil establecer categorías genéricas para las aplicaciones del software que sean significativas. Conforme aumenta la complejidad del software, es más difícil establecer compartimentos nítidamente separados. Las siguientes áreas del software indican la amplitud de las aplicaciones potenciales:

Software de sistemas. El software de sistemas es un conjunto de programas que han sido escritos para servir a otros programas. Algunos programas de sistemas (por ejemplo: compiladores, editores y utilidades de gestión de archivos) procesan estructuras de información complejas pero determinadas. Otras aplicaciones de sistemas (por ejemplo: ciertos componentes del sistema operativo, utilidades de manejo de periféricos, procesadores de telecomunicaciones) procesan datos en gran medida indeterminados. En cualquier caso, el área del software de sistemas se caracteriza por una fuerte interacción con el hardware de la computadora; una gran utilización por múltiples usuarios; una operación concurrente que requiere una planificación, una compartición de recursos y una sofisticada gestión de procesos; unas estructuras de datos complejas y múltiples interfaces externas.

Software de tiempo real. El software que coordina/analiza/controla sucesos del mundo real conforme ocurren, se denomina de tiempo real. Entre los elementos del software de tiempo real se incluyen: un componente de adquisición de datos que recolecta y da formato a la información recibida del entorno externo, un componente de análisis que transforma la información según lo requiera la aplicación, un componente de control/salida que responda al entorno externo, y un componente de monitorización que coordina todos los demás componentes, de forma que pueda mantenerse la repuesta en tiempo real (típicamente en el rango de un milisegundo a un segundo).

Software de gestión. El proceso de la información comercial constituye la mayor de las áreas de aplicación del software. Los «sistemas» discretos (por ejemplo: nóminas, cuentas de haberes-débitos, inventarios, etc.) han evolucionado hacia el software de sistemas de información de gestión **(SIG)** que accede a una o más bases de datos que contienen información comercial. Las aplicaciones en esta área reestructuran los datos existentes para facilitar las operaciones comerciales o gestionar la toma de decisiones. Además de las tareas convencionales de procesamientos de datos, las aplicaciones de software de gestión también realizan cálculo interactivo (por ejemplo: el procesamiento de transacciones en puntos de ventas).

Software de ingeniería y científíco. El software de ingeniería y científico está caracterizado por los algoritmos de «manejo de números». Las aplicaciones van desde la astronomía a la vulcanología, desde el análisis de la presión de los automotores a la dinámica orbital de las lanzaderas espaciales y desde la biología molecular a la fabricación automática. Sin embargo, las nuevas aplicaciones del área de ingeniería/ciencia se han alejado de los algoritmos convencionales numéricos. El diseño asistido por computadora (del inglés CAD), la simulación de sistemas y otras aplicaciones interactivas, han comenzado a coger características del software de tiempo real e incluso del software de sistemas.

Software empotrado. Los productos inteligentes se han convertido en algo común en casi todos los mercados de consumo e industriales. El software empotrado reside en memoria de sólo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumo. El software empotrado puede ejecutar funciones muy limitadas y curiosas (por ejemplo: el control de las teclas de un horno de microondas) o suministrar una función significativa y con capacidad de control (por ejemplo: funciones digitales en un automóvil, tales como control de la gasolina, indicadores en el salpicadero, sistemas de frenado, etc.).

Referencia Web

Se puede encontrar una de las mayores bibliotecas de *shareware/freeware* en **www.shareware.com**

Software de computadoras personales. El mercado del software de computadoras personales ha germinado en las pasadas dos décadas. El procesamiento de textos, las hojas de cálculo, los gráficos por computadora, multimedia, entretenimientos, gestión de bases de datos, aplicaciones financieras, de negocios y personales y redes o acceso a bases de datos externas son algunas de los cientos de aplicaciones.

Software basado en Web. Las páginas Web buscadas por un explorador son software que incorpora instrucciones ejecutables (por ejemplo, CGI, HTML, Perl, o Java), y datos (por ejemplo, hipertexto y una variedad de formatos de audio y visuales). En esencia, la red viene a ser una gran computadora que proporciona un recurso software casi ilimitado que puede ser accedido por cualquiera con un modem.

Software de inteligencia artificial. El software de inteligencia artificial (IA) hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados el cálculo o el análisis directo. Los sistemas expertos, también llamados sistemas basados en el conocimiento, reconocimiento de patrones (imágenes y voz), redes neuronales artificiales, prueba de teoremas, y los juegos son representativos de las aplicaciones de esta categoría.

1.3 SOFTWARE: ¿UNA CRISIS EN EL HORIZONTE?

Muchos observadores de la industria (incluyendo este autor) han caracterizado los problemas asociados con el desarrollo del software como una «crisis». Más de unos cuantos libros (por ejemplo: [GLA97], [FLO97], [YOU98a]) han recogido el impacto de algunos de los fallos mas importantes que ocurrieron durante la década pasada. No obstante, los mayores éxitos conseguidos por la industria del software han llevado a preguntarse si el término (crisis del software) es aún apropiado. Robert Glass, autor de varios libros sobre fallos del software, representa a aquellos que han sufrido un cambio de pensamiento. Expone [GLA98]: «Puedo ver en mis ensayos históricos de fallos y en mis informes de excepción, fallos importantes en medio de muchos éxitos, una copa que está [ahora] prácticamente llena.»

> **Cita:**
> La mayoría de los expertos están de acuerdo en que la manera más probable para que el mundo se destruya es por accidente. Ahí es donde nosotros entramos; somos profesionales de la informática, provocamos accidentes.
> **Nathaniel Borenstein**

La palabra crisis se define en el *diccionario Webster* como «un punto decisivo en el curso de algo, momento, etapa o evento decisivo o crucial». Sin embargo, en términos de calidad del software total y de velocidad con la cual son desarrollados los productos y los sistemas basados en computadoras, no ha habido ningún «punto crucial», ningún «momento decisivo», solamente un lento cambio evolutivo, puntualizado por cambios tecnológicos explosivos en las disciplinas relacionadas con el software.

Cualquiera que busque la palabra *crisis* en el diccionario encontrará otra definición: «el punto decisivo en el curso de una enfermedad, cuando se ve más claro si el paciente vivirá o morirá». Esta definición puede darnos una pista sobre la verdadera naturaleza de los problemas que han acosado el desarrollo del software.

Lo que realmente tenemos es una aflicción crónica[1]. La palabra *aflicción* se define como «algo que causa pena o desastre». Pero la clave de nuestro argumento es la definición del adjetivo *crónica:* «muy duradero *o* que reaparece con frecuencia continuando indefinidamente». Es bastante más preciso describir los problemas que hemos estado aguantando en el negocio del software como una aflicción crónica, en vez de como una crisis.

Sin tener en cuenta como lo llamemos, el conjunto de problemas encontrados en el desarrollo del software de computadoras no se limitan al software que «no funciona correctamente». Es más, el mal abarca los problemas asociados a cómo desarrollar software, cómo mantener el volumen cada vez mayor de software existente y cómo poder esperar mantenemos al corriente de la demanda creciente de software.

Vivimos con esta aflicción desde este día —de hecho, la industria prospera a pesar de ello—. Y así, las cosas podrán ser mejores si podemos encontrar y aplicar un remedio.

1.4 MITOS DEL SOFTWARE

Muchas de las causas de la crisis del software se pueden encontrar en una mitología que surge durante los primeros años del desarrollo del software. A diferencia de los mitos antiguos, que a menudo proporcionaban a los hombres lecciones dignas de tener en cuenta, los mitos del software propagaron información errónea y confusión. Los mitos del software tienen varios atributos que los hacen insidiosos: por ejemplo, aparecieron como declaraciones razonables de hechos (algunas veces conteniendo elementos verdaderos), tuvieron un sentido intuitivo y frecuentemente fueron promulgados por expertos que «estaban al día».

[1] Esta terminología fue sugerida por el profesor Daniel Tiechrow de la Universidad de Michigan en una conferencia impartida en Ginebra, Suiza, Abril, 1989.

Mitos de gestión. Los gestores con responsabilidad sobre el software, como los gestores en la mayoría de las disciplinas, están normalmente bajo la presión de cumplir los presupuestos, hacer que no se retrase el proyecto y mejorar la calidad. Igual que se agarra al vacío una persona que se ahoga, un gestor de software se agarra frecuentemente a un mito del software, aunque tal creencia sólo disminuya la presión temporalmente.

Mito. Tenemos ya un libro que está lleno de estándares y procedimientos para construir software, ¿no le proporciona ya a mi gente todo lo que necesita saber?

Realidad. Está muy bien que el libro exista, pero jse usa?.¿conocen los trabajadores su existencia?, jrefleja las prácticas modernas de desarrollo de software?, ¿es completo?, ¿está diseñado para mejorar el tiempo de entrega mientras mantiene un enfoque de calidad? En muchos casos, la respuesta a todas estas preguntas es «no».

Cita:
En ausencia de estándares significativos, una nueva industria como la del software pasa a depender de las costumbres.
Tom De Marco

Mito. Mi gente dispone de las herramientas de desarrollo de software más avanzadas, después de todo, les compramos las computadoras más modernas.

Realidad. Se necesita mucho más que el último modelo de computadora grande o de PC para hacer desarrollo de software de gran calidad. Las herramientas de ingeniería del software asistida por computadora (CASE) son más importantes que el hardware para conseguir buena calidad y productividad, aunque la mayoría de los desarrolladores del software todavía no las utilicen eficazmente.

Mito. Si fallamos en la planificación, podemos añadir más programadores y adelantar el tiempo perdido (el llamado algunas veces «concepto de la horda Mongoliana»).

Realidad. El desarrollo de software no es un proceso mecánico como la fabricación. En palabras de Brooks [BRO75]: «...añadir gente a un proyecto de software retrasado lo retrasa aún más». Al principio, esta declaración puede parecer un contrasentido. Sin embargo, cuando se añaden nuevas personas, la necesidad de aprender y comunicarse con el equipo puede y hace que se reduzca la cantidad de tiempo gastado en el desarrollo productivo. Puede añadirse gente, pero sólo de una manera planificada y bien coordinada.

Referencia Web
la red de gestión de proyectos de software en **www.spmn.com** puede ayudarle a desmitificar estos y otros mitos.

Mitos del Cliente. Un cliente que solicita una aplicación de software puede ser una persona del despacho de al lado, un grupo técnico de la sala de abajo, el departamento de ventas o una compañía exterior que solicita un software bajo contrato. En muchos casos, el cliente cree en los mitos que existen sobre el software, debido a que los gestores y desarrolladores del software hacen muy poco para corregir la mala información. Los mitos conducen a que el cliente se cree una falsa expectativa y, finalmente, quede insatisfecho con el que desarrolla el software.

Mito. Una declaración general de los objetivos es suficiente para comenzar a escribir los programas —podemos dar los detalles más adelante—.

Realidad. Una mala definición inicial es la principal causa del trabajo baldío en software. Es esencial una descripción formal y detallada del ámbito de la información, funciones, comportamiento, rendimiento, interfaces, ligaduras del diseño y criterios de validación. Estas características pueden determinarse sólo después de una exhaustiva comunicación entre el cliente y el analista.

Mito. Los requisitos del proyecto cambian continuamente, pero los cambios pueden acomodarse fácilmente, ya que el software es flexible.

Referencia cruzada
La gestión y control de cambio está tratada con detalle en el Capítulo 9

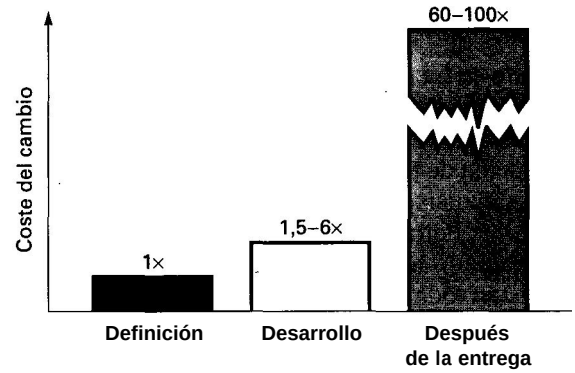


FIGURA 1.3. El impacto del cambio.

Realidad. Es verdad que los requisitos del software cambian, pero el impacto del cambio varía según el momento en que se introduzca. La Figura **1.3** ilustra el impacto de los cambios. Si se pone cuidado al dar la definición inicial, los cambios solicitados al principio pueden acomodarse fácilmente. El cliente puede revisar los requisitos y recomendar las modificaciones con relativamente poco impacto en el coste. Cuando los cambios se solicitan durante el diseño del software, el impacto en el coste crece rápidamente. Ya se han acordado los recursos a utilizar y se ha establecido un marco de trabajo del diseño. Los cambios pueden producir trastornos que requieran recursos adicionales e importantes modificaciones del diseño; es decir, coste adicional. Los

cambios en la función, rendimiento, interfaces u otras características, durante la implementación (codificación y prueba) pueden tener un impacto importante sobre el coste. Cuando se solicitan al final de un proyecto, los cambios pueden producir un orden de magnitud más caro que el mismo cambio pedido al principio.

Mitos de los desarrolladores. Los mitos en los que aún creen muchos desarrolladores se han ido fomentando durante 50 años de cultura informática. Durante los primeros días del desarrollo del software, la programación se veía como un arte. Las viejas formas y actitudes tardan en morir.

Mito. Una vez que escribimos el programa y hacemos que funcione, nuestro trabajo ha terminado.

Realidad. Alguien dijo una vez: «cuanto más pronto se comience a escribir código, más se tardará en terminarlo». Los datos industriales [LIE80, JON91, PUT97] indican que entre el 60 y el 80 por ciento de todo el esfuerzo dedicado a un programa se realizará después de que se le haya entregado al cliente por primera vez.

Trabaja muy duro poro entender lo que tienes que hacer antes de empezar. No serías copoz de desarrollar codo detalle; por más que sepas, tomo el menor riesgo.

Mito. Hasta que no tengo el programa «ejecutándose», realmente no tengo forma de comprobar su calidad.

Realidad. Desde el principio del proyecto se puede aplicar uno de los mecanismos más efectivos para garantizar la calidad del software: *la revisión técnicaformal.* La revisión del software (descrito en el Capítulo 8) es un «filtro de calidad» que se ha comprobado que es más efectivo que la prueba, para encontrar ciertas clases de defectos en el software.

Mito. Lo único que se entrega al terminar el proyecto es el programa funcionando.

Realidad. Un programa que funciona es sólo una parte de una *configuración del software* que incluye muchos elementos. La documentación proporciona el fundamento para un buen desarrollo y, lo que es más importante, proporciona guías para la tarea de mantenimiento del software.

Muchos profesionales del software reconocen la falacia de los mitos descritos anteriormente. Lamentablemente, las actitudes y métodos habituales fomentan una pobre gestión y malas prácticas técnicas, incluso cuando la realidad dicta un método mejor. El reconocimiento de las realidades del software es el primer paso hacia la formulación de soluciones prácticas para su desarrollo.

RESUMEN

El software se ha convertido en el elemento clave de la evolución de los sistemas y productos informáticos. En los pasados 50 años, el software ha pasado de ser una resolución de problemas especializada y una herramienta de análisis de información, a ser una industria por **sí** misma. Pero la temprana cultura e historia de la «programación» ha creado un conjunto de problemas que persisten todavía hoy. El software se ha convertido en un factor que limita la evolución de los sistemas informáticos. El software se compone de programas, datos y documentos. Cada uno de estos elementos componen una configuración que se crea como parte del proceso de la ingeniería del software. El intento de la ingeniería del software es proporcionar un marco de trabajo para construir software con mayor calidad.

Cuando te pones o pensar, no encuentras tiempo poro la disciplino de lo ingeniería del software, y te preguntas: «¿Tendré tiempo para poder hacerlo?»

REFERENCIAS

[BRO75] Brooks, F., *The Mytical Man-Month,* Addison-Wesley, 1975.

[DEJ98] De Jager, P., et al, *Countdown Y2K: Business Survival Planning for the Year 2000,* Wiley, 1998.

[DEM95] De Marco, T., *Why Does Software Cost So Much?,* Dorset House, 1995, p. 9.

[FEI83] Feigenbaum, E. A., y P. McCorduck, *The Fith Generation,* Addison-Wesley, 1983.

[FLO97] Flowers, S., *Software Failure, Management Failure-Amaicing Stories and Cautionary Tails,* Wiley, 1997 (?).

[GLA97] Glass, R. L., *Software Runaways,* Prentice Hall, 1997.

[GLA98] Glass, R. L.,«Is there Really a Software Crisis?», *IEEE Software,* vol. 15, n.º 1, Enero 1998, pp. 104-105.

[HAM93] Hammer, M., y J. Champy, *Reengineering the Corporation,* HarpperCollins Publisher, 1993.

[JON91] Jones, C., *Applied Software Measurement,* McGraw-Hill, 1991.

[KAR99] Karlson, E., y J. Kolber, *A Basic Introduction to Y2K:* How *the Year 2000 Computer Crisis Affects You?,* Next Era Publications, Inc., 1999.

[LEV95] Levy, S., «The Luddites Are Pack», *Newsweek*, 12 de Julio de 1995, p. 55.

[LEV99] Levy, S., «The New Digital Galaxy», *Newsweek*, 31 de Mayo de 1999, p.57.

[LIE80] Lientz, B., y E. Swanson, *Software Maintenance Management*, Addison Wesley, 1980.

[NAI82] Naisbitt, J., *Megatoends*, Warner Books, 1982.

[NOR98] Norman, D., *The Invisible Computer*, MIT Press, 1998.

[OSB79] Osborne, A., *Running Wild-The Next Industrial Revolution*, Osborne/McGraw-Hill, 1979.

[PUT97] Putnam, L., y W. Myers, *Industrial Strength Software*, IEEE Computer Society Press, 1997.

[STO89] Stoll, C., *The cuckoo's Egg*, Doubleday, 1989.

[TOF80] Toffler, A., *The Third Wave*, Morrow Publishers, 1980.

[TOF90] Toffler, A., *Powershift*, Bantam Publishers, 1990.

[YOU92] Yourdon, E., *The Decline and Fall of the American Programmer*, Yourdon Press, 1992.

[YOU96] Yourdon, E., *The Rise and Resurrection of the American Programmer*, Yourdon Press, 1996.

[YOU98a] Yourdon, E., *Death March Projects*, Prentice-Hall, 1998.

[YOU98b] Yourdon, E., y J. Yourdon, *Time Bomb* 2000, Prentice-Hall, 1998.

PROBLEMAS Y PUNTOS A CONSIDERAR

1.1. El software es la característica que diferencia a muchos productos y sistemas informáticos. Dé ejemplos de dos o tres productos y de, al menos, un sistema en el que el software, no el hardware, sea el elemento diferenciador.

1.2. En los años cincuenta y sesenta la programación de computadoras era un arte aprendido en un entorno básicamente experimental. ¿Cómo ha afectado esto a las prácticas de desarrollo del software hoy?

1.3. Muchos autores han tratado el impacto de la «era de la información». Dé varios ejemplos (positivos y negativos) que indiquen el impacto del software en nuestra sociedad. Repase algunas referencias de la Sección 1.1 previas a 1990 e indique dónde las predicciones del autor fueron correctas y dónde no lo fueron.

1.4. Seleccione una aplicación específica e indique: (a) la categoría de la aplicación de software (Sección 1.2.2) en la que encaje; (b) el contenido de los datos asociados con la aplicación; (c) la información determinada de la aplicación.

1.5. A medida que el software se difunde más, los riesgos para el público (debido a programas defectuosos) se convierten en una preocupación cada vez más significativa. Desarrolle un escenario realista del juicio final (distinto a Y2K) en donde el fallo de computadora podría hacer un gran daño (económico o humano).

1.6. Lea detenidamente el grupo de noticias de Internet **comp.risk** y prepare un resumen de riesgos para las personas con las que se hayan tratado Últimamente. Código alternativo: *Software Engineering Notes* publicado por la ACM.

1.7. Escriba un papel que resuma las ventajas recientes en una de las áreas de aplicaciones de software principales. Entre las selecciones potenciales se incluyen: aplicaciones avanzadas basadas en Web, realidad virtual, redes neuronales artificiales, interfaces humanas avanzadas y agentes inteligentes.

1.8. Los mitos destacados en la Sección 1.4 se están viniendo abajo lentamente a medida que pasan los años. Pero otros se están haciendo un lugar. Intente añadir un mito o dos mitos «nuevos» a cada categoría.

OTRAS LECTURAS Y FUENTES DE INFORMACIÓN

Literalmente existen miles de libros escritos sobre software de computadora. La gran mayoría tratan los lenguajes de programación o aplicaciones de software, y sólo unos pocos tratan el software en sí. Pressman y Herron *(Software Sock*, Dorset House, 1991) presentaron una discusión (dirigida a no profesionales) acerca del software y del modo en que lo construyen los profesionales.

El libro, éxito de ventas, de Negroponte *(Being Digital*, Alfred A. Knopf, Inc., 1995) proporciona una visión de las computadoras y de su impacto global en el siglo XXI. Los libros de Norman [NOR98] y Bergman *(Information Appliances & Beyond*, Academic Pres/Morgan Kauffman, 2000) sugieren que el impacto extendido del PC declinará al mismo tiempo que las aplicaciones de información y la difusión de la programación conecten a todos en el mundo industrializado y casi todas las aplicaciones a la nueva infraestructura de Internet.

Minasi *(The Software Conspiracy: Why Software Companies Put Out Faulty Products, How They Can Hurt* ***You,*** *and What You Can Do*, McGraw-Hill, 2000) argumentó que el «azote moderno» de los errores del software puede eliminarse y sugiere formas para hacerlo. DeMarco (*Why Does Software Cost So Much?*, Dorset House, 1995) ha producido una colección de ensayos divertidos e interesantes sobre el software y el proceso a través del cual se desarrolla.

En Intemet están disponibles una gran variedad de fuentes de información relacionadas con temas de gestión y de software. Se puede encontrar una lista actualizada con referencias a sitios (páginas) web relevantes en **http://www.pressman5.com.**

2.4. El modelo del caos sugiere que un bucle de resolución de problemas se pueda aplicar en cualquier grado de resolución. Estudie la forma en que se aplicaría el bucle (1) para comprender los requisitos de un producto de tratamiento de texto; (2) para desarrollar un componente de corrección ortográfica y gramática avanzado para el procesador de texto; (3) para generar el código para un módulo de programa que determine el sujeto, predicado y objeto de una oración en inglés.

2.5. ¿Qué paradigmas de ingeniería del software de los presentados en este capítulo piensa que sería el más eficaz? ¿Por qué?

2.6. Proporcione cinco ejemplos de proyectos de desarrollo del software que sean adecuados para construir prototipos. Nombre dos o tres aplicaciones que fueran más difíciles para construir prototipos.

2.7. El modelo DRA a menudo se une a herramientas CASE. Investigue la literatura y proporcione un resumen de una herramienta típica CASE que soporte DRA.

2.8. Proponga un proyecto específicode software que sea adecuado para el modelo incremental. Presente un escenario para aplicar el modelo al software.

2.9. A medida que vaya hacia afuera por el modelo en espiral, ¿qué puede decir del software que se está desarrollando o manteniendo?

2.10. Muchas personas creen que la Única forma en la que se van a lograr mejoras de magnitud en la calidad del software y en su productividad es a través del desarrollo basado en componentes. Encuentre tres o cuatro artículos recientes sobre el asunto y resúmalos para la clase.

2.11. Describa el modelo de desarrollo concurrente con sus propias palabras.

2.12. Proporcione tres ejemplos de técnicas de cuarta generación.

2.13. ¿Qué es más importante, el producto o el proceso?

OTRAS LECTURAS Y FUENTES DE INFORMACIÓN

El estado actual del arte en la ingeniería del software se puede determinar mejor a partir de publicaciones mensuales tales como ***IEEE Software, Computer e IEEE Transactions on Software Engineering.*** Periódicos sobre la industria tales como *Application Developement Trends, Cutter IT Journal* y ***Software Developement*** a menudo contienen artículos sobre temas de ingeniería del software. La disciplina se «resume» cada año en el *Proceeding of the International Conference on Software Engineering,* promocionado por el IEEE y ACM y tratado en profundidad en periódicos como ***ACM Transactions on Software Engineering and Methodology, ACM Software Engineering Notes*** y *Annals of Software Engineering.*

Se han publicado muchos libros de ingeniería del software en los últimos años. Algunos presentan una visión general del proceso entero mientras que otros se dedican a unos pocos temas importantes para la exclusión de otros. Las siguientes son tres antologías que abarcan algunos temas importantes de ingeniería del software:

Keyes, J. (ed.), *Software **Engineering Productivity** Handbook,* McGraw-Hill, 1993.

McDermid, J. (ed.), ***Software** Engineer's **Reference Book,*** CRC Press, 1993.

Marchiniak, J.J. (ed.), ***Encyclopedia** of **Software** Engineering,* Wiley, 1994.

Gautier ***(Distributed Engineering** of **Software,*** Prentice-Hall, 1996) proporciona sugerencias y directrices para organizaciones que deban desarrollar software en lugares geográficamente distantes.

En la parte más brillante del libro de Robert Glass ***(Software Conflict,*** Yourdon Press, 1991) se presentan ensayos controvertidos y amenos sobre el software y el proceso de la ingeniería del software. Pressman y Herron ***(Software Shock,*** Dorset House, 1991) consideran el software y su impacto sobre particulares, empresas y el gobierno.

El Instituto de ingeniería del software (IIS localizado en la Universidad de Carnegie-Mellon) ha sido creado con la responsabilidad de promocionar series monográficas sobre la ingeniería del software. Los profesionales académicos, en la industria y el gobierno están contribuyendo con nuevos trabajos importantes. El Consorcio de Productividad del Software dirige una investigación adicional de ingeniería de software.

En la última década se ha publicado una gran variedad de estándares y procedimientos de ingeniería del software. El ***IEEE Software Engineering Standards*** contiene muchos estándares diferentes que abarcan casi todos los aspectos importantes de la tecnología. Las directrices ISO 9000-3 proporcionan una guía para las organizaciones de software que requieren un registro en el estándar de calidad ISO 9001. Otros estándares de ingeniería del software se pueden obtener desde el MOD, la Autoridad Civil de Aviación y otras agencias del gobierno y sin ánimo de lucro. Fairclough ***(Software Engineering Guides,*** Prentice-Hall, 1996) proporciona una referencia detallada de los estándares de ingeniería del software producidos por European Space Agency (ESA).

En internet se dispone de una gran variedad de fuentes de información sobre la ingeniería del software y del proceso de software. Se puede encontrar una lista actualizada con referencias a sitios (páginas) web que son relevantes para el proceso del software en **http://www.pressman5.com.**