Capítulo 1

Introducción

1.1. Introducción

El software es una parte principal del entorno humano. La cantidad de aparatos de todo tipo que rodean a las personas, que se usan a diario y sin los que cada vez la vida sería más difícil de imaginar, que están controlados por un programa, por un software que rige su comportamiento es enorme.

Teléfonos, electrodomésticos, coches o cualquier vehículo que circule por una carretera, aviones, barcos, trenes, el aire acondicionado de la casa u oficina, los sistemas de control de edificios, aeropuertos o estaciones, las televisiones, los sistemas de gestión de las empresas, los robots de las fábricas de ensamblaje, la lista es interminable. Todos estos sistemas disponen de uno o más computadores, que constituyen el "hardware" del sistema, y de los programas que gobiernan su funcionamiento, que componen el "software" de los mismos.

El software está presente no sólo en los sistemas informáticos que realizan tareas de tratamiento de información, sino en un sinfín de sistemas de la más diversa complejidad. Son miles, millones de líneas de código que diariamente se programan para conseguir que todos estos sistemas funcionen como se desea. Esta tarea de construir el software la realizan los programadores, los cuales tienen que a su vez dar mantenimiento durante, en la mayoría de los casos, largo tiempo.

La ingeniería es, según la Real Academia de la Lengua, el conjunto de conocimientos y técnicas que permiten aplicar el saber científico a la utilización de la materia y de las fuentes de energía. Se pueden encontrar un sinfín de definiciones de ingeniería, en las que el denominador común es la aplicación práctica del conocimiento para la elaboración de cualquier tipo de producto o servicio.

En este libro se introduce el conjunto de técnicas y procedimientos que se han ido desarrollando a lo largo de las últimas décadas, para poder elaborar de una forma ordenada y eficiente tantas y tantas líneas de código que componen el software. Todas estas técnicas y procedimientos componen la ingeniería de software, una todavía joven rama de la ciencia.

1.2. Objetivos

En este capítulo se da una visión inicial de lo que es el software y cómo se produce. Igualmente se profundiza en el concepto de ingeniería del software. Se pretende que el lector adquiera una idea clara de los siguientes conceptos:

- Definición de software y requisitos de calidad exigibles.
- No todo el software es igual, según su aplicación y modo de desarrollarlo puede ser muy diverso. Se da una visión de los distintos tipos de software.
- Definición del concepto de Ingeniería del software y comprensión de su origen.
- Existen algunas ideas preestablecidas acerca del software que se analizan.

1.3. ¿Qué es el software?

En un sistema informático el hardware se identifica con facilidad, son los aparatos físicos. El software, sin embargo, es algo más difícil de caracterizar, y a veces se define por exclusión: el software es todo lo que no es hardware. El software incluye, por supuesto, los programas que gobiernan el funcionamiento del sistema, pero también incluye otros elementos tales como documentos, bases de datos, o algo tan inmaterial como son los procedimientos de operación o de mantenimiento periódico.

El software puede ser en sí mismo un producto que se venda, por ejemplo un procesador de textos o un programa de tratamiento de imágenes, o tan sólo una parte, en la mayoría de los casos esencial, de un producto más complejo, por ejemplo el programa que gobierna la inyección de gasoil en un motor diésel, o puede ser el medio para dar un servicio, por ejemplo el programa que permite realizar una transferencia bancaria. Lo que es indudable es que la elaboración del software ocupa a millones de personas en todo el mundo y se puede considerar una actividad económica en sí misma.

1.3.1. Calidad del software

La calidad de un producto puede valorarse desde puntos de vista diversos. El software no es una excepción, y existen por tanto diferentes enfoques para la valoración de su calidad.

Lo ideal sería poder medir la calidad del software como se miden ciertos aspectos de calidad de otros productos de ingeniería: pureza de un producto, resistencia de un material, sus dimensiones, etcétera. Desgraciadamente esto no resulta fácil, y las técnicas de aplicación de métricas precisas a los productos software están todavía en evolución.

Existe un esquema general de mediciones de la calidad de software propuesto por MacCall y otros [McCall78], basado en valoraciones a tres niveles diferentes, denominados factores, criterios y métricas. Los factores de calidad constituyen el nivel superior, y son la valoración propiamente dicha, significativa, de la calidad. Esta valoración no se hace directamente, sino en función de ciertos criterios o aspectos de nivel intermedio que influyen en los factores de calidad. Las métricas están en el nivel inferior, son mediciones puntuales de determinados atributos o características del producto, y son la base para evaluar los criterios intermedios. Entre los factores de calidad propuestos encontramos los siguientes:

- CORRECCIÓN. Es el grado en que un producto software cumple con sus especificaciones. Podría estimarse como el porcentaje de requisitos que se cumplen adecuadamente.
- FIABILIDAD. Es el grado de ausencia de fallos durante la operación del producto software. Puede estimarse como el número de fallos producidos o el tiempo durante el que permanece inutilizable durante un intervalo de operación dado.
- EFICIENCIA. Es la relación entre la cantidad de resultados suministrados y los recursos requeridos durante la operación. Se mediría como la inversa de los recursos consumidos para realizar una operación dada.
- SEGURIDAD. Es la dificultad para el acceso a los datos o a los datos o a las operaciones por parte de personal no autorizado.
- FACILIDAD DE USO. Es la inversa del esfuerzo requerido para aprender a usar un producto software y utilizarlo adecuadamente.
- MANTENIBILIDAD. Es la facilidad para corregir el producto en caso necesario. Se aplica propiamente el mantenimiento correctivo.
- FLEXIBILIDAD. Es la facilidad para modificar el producto software. Se aplica propiamente al mantenimiento adaptativo y al perfectivo.

- FACILIDAD DE PRUEBA. Es la inversa del esfuerzo requerido para ensayar un producto software y comprobar su corrección o fiabilidad.

- TRANSPORTABILIDAD. Es la facilidad para adaptar el producto software a una plataforma (hardware + sistema operativo) diferente de aquella para la que se desarrolló inicialmente.

- REUSABILIDAD. Es la facilidad para emplear partes del producto en otros desarrollos posteriores. Se facilita mediante una adecuada organización de los módulos y funciones durante el diseño.

- INTEROPERATIVIDAD. Es la facilidad o capacidad del producto software para trabajar en combinación con otros productos.

Estos factores de calidad se centran en características del producto software. En muchos casos se contemplan también otros aspectos relativos al proceso de desarrollo, ya que la organización de éste influye muy directamente en la calidad del producto obtenido.

Comprobar la calidad de un software es una tarea compleja. Las pruebas o ensayos consisten en hacer un producto software o una parte de él en condiciones determinadas, y comprobar si los resultados son correctos. El objetivo de las pruebas es descubrir los errores que pueda contener el software ensayado.

Las pruebas no permiten garantizar la calidad de un producto. Puede decirse que una prueba tiene éxito si se descubre algún error, con lo que se sabe que el producto no cumple con algún criterio de calidad. Por el contrario, si la prueba no descubre ningún error, no se garantiza con ello la calidad del producto, ya que pueden existir otros errores que habrían de descubrirse mediante pruebas diferentes. Esto es así porque nunca puede ensayarse un producto de forma exhaustiva, sino que las pruebas sólo hacen que el producto realice una parte ínfima de la enorme variedad de operaciones concretas que podría realizar.

En la figura 1.1 podemos observar de forma simplificada la evolución de la tasa de fallos de un software en el tiempo. Inicialmente esta tasa es muy alta. Según se van corrigiendo los errores se reduce rápidamente. Sin embargo, a lo largo de la vida de un software es frecuente realizar mejoras funcionales, dando lugar a distintas versiones. Cada vez que introducimos cambios en las nuevas versiones, el número de errores del software se dispara, haciendo de nuevo necesario la corrección de los mismos.

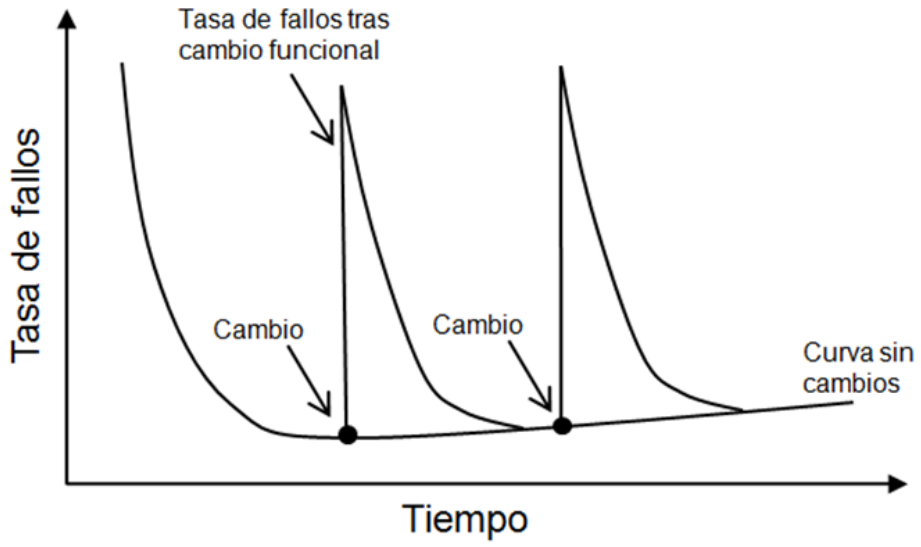

Figura 1.1: Evolución de fallos en un Sistema Software

1.3.2. Tipos de software

Clasificar el software no es una tarea fácil debido a la gran variedad de aplicaciones y métodos de desarrollo que existe. Una de las clasificaciones más completas se encuentra en [Pressman10], donde se agrupa el software en siete grandes categorías:

- SOFTWARE DE SISTEMAS

  Lo forman todos aquellos programas necesarios para dar soporte a otros programas, como los sistemas operativos, los compiladores o los programas de gestión de redes. Su principal característica es su alto grado de interacción con el hardware, ya que en muchos casos deben gestionar de forma eficiente el acceso al hardware por parte de otros programas o usuarios.

- SOFTWARE DE APLICACIÓN

  Son aplicaciones desarrolladas para resolver problemas específicos de los negocios. En esta categoría incluiríamos el software de gestión de los bancos o de las grandes empresas en general, como los ERP (Enterprise Resource Planning).

- SOFTWARE DE INGENERÍA Y CIENCIAS

  El objetivo es la programación de elaborados algoritmos matemáticos para modelar y simular complejos sistemas o procesos, tales como reacciones nucleares, modelos metereológicos, la red eléctrica de un país o el diseño de un avión.

- SOFTWARE INCRUSTADO

  Reside en el interior de un producto o sistema, y su objetivo es controlarlo, definir su comportamiento. Suele ser muy específico y de pequeñas dimensiones, con la necesidad de operar en tiempo real. Desde el regulador de temperatura de una vivienda hasta el sistema de frenos de un vehículo, están gobernados por este tipo de software.

- SOFTWARE DE LINEA DE PRODUCTO

  Su objetivo es dar una determinada funcionalidad al consumidor. En esta categoría encontramos procesadores de texto, hojas de cálculo o las aplicaciones de contabilidad para pequeñas empresas.

- APLICACIONES WEB ("WEBAPPS")

  En los últimos años se ha extendido su utilización con la generalización de los aparatos móviles con acceso a redes. Inicialmente simplemente se componían de archivos de hipertexto para la presentación de información, sin embargo hoy día tienen capacidad de cómputo y está integradas con aplicaciones y bases de datos corporativas. A través de ellas se puede operar una cuenta bancaria, realizar todo tipo de compras, utilizar juegos muy elaborados ó conocer el tiempo en cualquier parte del mundo. La comodidad, rapidez y vistosidad son determinantes a la hora de que tengan éxito.

- SOFTWARE DE INTELIGENCIA ARTIFICIAL.

  Incluye aplicaciones de robótica, visión artificial, redes neuronales o sobre la teoría de juegos. Utilizan algoritmos no numéricos para la resolución de los problemas, como por ejemplo árboles lógicos de búsqueda.

1.4. ¿Cómo se fabrica el software?

Antes de la revolución industrial, los productos realizaban de forma artesanal. El artesano aprendía una serie de técnicas y procedimientos que le permitían elaborarlos manualmente. Dichas técnicas y procedimientos rara vez se documentaban, y se transmitían de maestro a aprendiz a lo largo de los años.

En el siglo XIX la industrialización y la producción en serie cambio radicalmente esta forma de hacer. Se fabricaban cientos y miles de productos todos iguales y se comercializaban masivamente. Para producir de forma eficiente se desarrollaron una serie de técnicas, procedimiento, estándares y normas, los cuales permitían tener perfectamente controlado todo el ciclo de vida del producto, desde que éste es una simple idea, hasta que se entrega al cliente final.

La elaboración de cualquier producto industrial conlleva un largo proceso, desde la concepción del producto pensando en la necesidad que se desea que cubra y su mercado potencial, hasta que sale terminado de la fábrica. Todo este proceso está concebido para lograr los objetivos de calidad y coste que se consideren necesarios para poder venderlo.

De igual forma, en las décadas iniciales de existencia de la informática, la labor de desarrollo de software se planteaba como una actividad artesanal, basada en la labor de personas habilidosas y más o menos creativas, que actuaban en forma individual y relativamente poco disciplinada.

Al aumentar la capacidad de los computadores gracias a los avances del hardware, aumentó también la complejidad de las aplicaciones a programar, y se apreció la necesidad de una mejor organización de la producción de software, basada en el trabajo en equipo, con la consiguiente división y organización del trabajo, y el empleo de herramientas apropiadas que automaticen las labores triviales y repetitivas.

La identificación formal del problema origina una frenética actividad para la creación de metodologías concretas de desarrollo y, en general, en la concepción de la ingeniería del software como disciplina. A finales de los años 60, se acuña el termino Ingeniería del Software en un congreso de la OTAN de manera formal. Con esta denominación se designa el empleo en el desarrollo del software, de técnicas y procedimientos típicos de la ingeniería en general.

El software tiene una particularidad especial frente a cualquier producto físico que podamos imaginar: una vez diseñado, este se puede replicar con tremenda facilidad, sin necesidad de un proceso de fabricación propiamente dicho. A pesar de ello, la ingeniería del software se ha desarrollado a partir de la ingeniería industrial.

La norma de calidad [ISO1694908] es un conjunto de normas de calidad y gestión de la calidad, adoptadas por la industria del automóvil como herramienta básica para desarrollar los procesos necesarios en la producción de un vehículo de forma competitiva y satisfactoria para el cliente.

En el capítulo 2 de este libro se analizan los procesos más importantes para la producción del software y diferentes modelos para organizarlos, de forma similar a los procesos mostrados en la figura 1.2 para la industria del automóvil.

Figura 1.2: Enfoque de calidad en la industria del automóvil

La ingeniería del software amplía la visión del desarrollo del software como una actividad esencialmente de programación, contemplando además otras actividades de análisis y diseño previos, y de integración y verificación posteriores. La distribución de todas estas actividades a lo largo del tiempo constituye los se ha dado en llamar ciclo de vida del desarrollo de software.

Concretando, se tomará la definición de ingeniería del software de [SWEBOK04]. Ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software, y el estudio de estos enfoques, es decir, la aplicación de la ingeniería al software.

A lo largo de los años 70 aparecen las herramientas CASE (Computer Aided Software Engineering) de diseño asistido por ordenador, que se aplican ampliamente durante los 80. Las herramientas CASE tradicionales apoyaban a las actividades anteriores a la programación o codificación, para la que se seguías empleando herra-

mientas tradicionales, como los compiladores, que funcionaban de forma totalmente separada de la herramienta CASE.

En los 90 se amplía el campo de aplicación de las herramientas de ingeniería de software, reconociendo la necesidad de automatizar aún más la labor de desarrollo mediante la formalización del proceso completo de producción de software y el empleo de herramientas que soporten todo el ciclo de vida de desarrollo. Estas herramientas se designan con las siglas IPSE (Integrated Project Support Environment) o, más recientemente, ICASE (Integrated CASE).

Con la llegada del siglo XXI, el desarrollo e implantación de Internet, muchos de los planteamientos de las herramientas CASE han evolucionado. Por un lado la posibilidad de distribuir el desarrollo de los proyectos en diferentes localizaciones y por otro la obligada necesidad de la utilización de esta plataforma como marco para el funcionamiento de la mayoría de las aplicaciones. Otro de los grandes retos que ha aparecido y triunfado recientemente son los "smartphones" o teléfonos inteligentes. Por el momento sólo se utilizan como plataformas de funcionamiento, pero no hay duda de que en breve formarán parte de las infraestructuras empleadas para el desarrollo del software.

A lo largo de estos períodos de tiempo fue surgiendo una importante comunidad científica en torno a la ingeniería de software. Dos organizaciones han liderado dicha comunidad, ACM e IEEE Computer Society, que han promovido activamente la puesta en práctica de esta disciplina. IEEE ha desarrollado la guía SWEBOK con el objeto de crear una acreditación para la profesión de ingeniero del software en Estados Unidos. Dicha guía da forma a los conocimientos necesarios para dominar esta disciplina y los diferenciales frente a otras relacionadas con el software, como las ciencias de la computación o la gestión de proyectos de software.

Todavía a día de hoy, la ingeniería de software se encuentra en una situación permanente de fuerte evolución, con avances continuos en las técnicas que resultan sin embargo insuficientes en cada momento. De hecho se discute si el término ingeniería del software es realmente válido, ya que una disciplina de ingeniería se caracteriza, en general, por haber alcanzado una cierta madurez en las técnicas a aplicar, basadas en un fundamento científico subyacente y no sólo en un pragmatismo artesanal.

1.5. Mitos del software

El continuo proceso de evolución en las disciplinas de desarrollo de software, junto con algunas de sus características particulares, tales como una relativa inmateriali-

dad,hacen difícil obtener una visión serena y justa de la ingeniería de software, provocando que por parte de los usuarios e incluso de algunos profesionales se mantengan opiniones infundadas sobre la importancia o influencia de determinados factores en el éxito o calidad de un producto software.

Alguna de estas opiniones, relativamente generalizadas, constituyen verdaderos mitos, difíciles de erradicar. Podemos mencionar, por ejemplo, las siguientes:

- El hardware es mucho más importante que el software. Manifiestamente falso, ya que al usar un computador nuestra interacción es fundamentalmente con el software, y sólo de una manera muy limitada el usuario accede directamente a elementos hardware del equipo. Este menosprecio por el software se evidencia en quienes consideran que la realización de copias "pirata" o ilegales de los programas no es una acción censurable.

- El software es fácil de desarrollar. Esto es falso para cualquier aplicación software de cierta importancia. El desarrollo de grandes sistemas es muy complejo y costoso, incluso aunque esos sistemas no empleen ningún material o hardware específico. De hecho el desarrollo de software exige una mayor proporción de manos de obra, frente al empleo de maquinaria, y por ello el progresivo aumento del costo de la mano de obra en los países desarrollados ha llevado a un crecimiento importante en el coste de los productos software.

- El software consiste exclusivamente en programas ejecutables: El software no se de esta manera. Al concebir un sistema informático de manera global hay que pensar en todos los elementos que intervienen: hardware, software y personas. Los procedimientos que han de seguir esas personas para usar correctamente el sistema son también elementos software. Igualmente es software toda la documentación del desarrollo, necesaria para poder mantener el producto después de haber sido puesto en explotación.

- El desarrollo de software es sólo una labor de programación: Falso, pues no se puede limitar el trabajo de desarrollo sólo a la fase de codificación. Las tareas de análisis y diseño no son meras actividades complementarias que puedan ser vistas como un costo añadido, necesario para organizar el trabajo en equipo. Las tareas de análisis y diseño son el fundamento para todo el resto del desarrollo, igual que el proyecto de un arquitecto o de un ingeniero es necesario para acometer la construcción de un edificio u otra obra pública, que no consiste simplemente en colocar materiales uno sobre otro.

- Es natural que el software contenga errores. No exactamente. Aunque es cierto que el desarrollo de software, como toda actividad humana, es susceptible de errores, no es admisible que los productos software siempre contengan errores. Si un producto hardware contiene defectos, se rechaza. Con el software debería ocurrir lo mismo. Por desgracia el software erróneo no puede simplemente sustituirse por otro sin defecto, ya que todas las copias del producto software son exactamente iguales. Los errores del software son errores durante su desarrollo inicial, y deben reducirse a un nivel tan bajo como en el diseño de los productos hardware durante la fase de desarrollo de ingeniería.

1.6. Conclusiones

El software lo componen el conjunto de programas que gobiernan el comportamiento de cualquier sistema basado en computador. En muchos casos el software tiene entidad en sí misma, y puede ser considerado un producto propiamente dicho.

La aplicación del conocimiento y del método científico a la elaboración del software, dan lugar a la disciplina que se conoce como ingeniería del software.

El ciclo de vida del software, igual que el de cualquier otro producto que se pueda elaborar, es la evolución del mismo desde el momento que se concibe hasta que se deja de utilizar.

1.7. Ejercicios propuestos

1. Piénsese en diez sistemas con los que se interactúe usted la vida diaria que estén controlados por algún tipo de programa. Hágase un listado e intente describir la funcionalidad de cada uno de ellos.

2. Supóngase el diagrama de las tasas de fallos de un coche en el tiempo. ¿Qué diferencias puede tener respecto a un producto de software?

3. Compárase el ciclo de vida de una lavadora con el de un programa de edición de texto. Elabórese un listado con las particularidades de cada uno de ellos.

4. Repásese las siete categorías de software y dar diez ejemplos de cada una de ellas.

5. Una práctica muy extendida para reutilizar software es "cortar y pegar" fragmentos de código. Esto puede llevar a que un trozo de código se encuentre

repetido muchas veces en un programa. ¿Cómo afecta esto al mantenimiento del programa? ¿Cómo se podrían solventarse los efectos negativos del código duplicado?

6. Los principales términos acuñados en este capítulo son software e ingeniería de software. Explíquense razonadamente las diferencias entre ambos términos.

7. Arguméntese la corrección o falsedad de las siguientes afirmaciones:

   - "Los requerimientos de software cambian continuamente, pero el cambio se asimila con facilidad debido a que el software es flexible".
   - "Una vez que se escriba el programa y se hace que funcione, el trabajo ha terminado"
   - "La ingeniería de software hará que se genere documentación voluminosa e innecesaria, e invariablemente retrasará el trabajo".