Universidad Nacional Autónoma de México.

Facultad de Ingeniería.

Ingeniería de Software

Grupo: 07

Profesor: Ing. Orlando Zaldívar Zamorategui

Equipo: 2

Tutorial IS 2025-1 4. Introducción a la ingeniería de Software. Estándares básicos. Introducción y aplicación del SWEBOK. Software Engineering Body of Knowledge.

Integrantes:

Calzada Pérez Daira Aimé - 319010532

Cisneros Paredes Diana Paola - 318690793

Marín Montaño Josué - 319235630

Martínez Rosales Hugo Armando - 319326095

Navarro Rodriguez Angel Efren - 319167959

Toledo Durán Jesús Rodrigo - 319124808

22/10/2024 **Ciudad Universitaria, CDMX, a 05/11/2024** **05/11/2024**

Índice

Tema

Introducción a la ingeniería de Software. Estándares básicos. Introducción y aplicación del SWEBOK. Software Engineering Body of Knowledge.

Temario

1. Objetivo

El propósito de este tutorial es ofrecer a los usuarios una comprensión sobre el SWEBOK (Software Engineering Body of Knowledge), ya que este documento es fundamental para establecer un marco de referencia sobre las mejores prácticas y principios en la ingeniería del software. Este tutorial tiene como objetivos:

- Presentar de manera clara las principales áreas que componen el SWEBOK..
- Describir las prácticas y técnicas esenciales dentro de cada área, enfatizando su importancia en el desarrollo de software eficaz y eficiente.
- Facilitar una comprensión intuitiva a través de diversos ejemplos.
- Fomentar la aplicación práctica de los conocimientos adquiridos fuera del entorno del tutorial, mediante preguntas que promuevan la reflexión y el análisis.

El objetivo final de este tutorial es que los usuarios no solo adquieran conocimientos teóricos sobre el SWEBOK, sino que también puedan aplicar estos conceptos en diversos contextos, promoviendo así un entendimiento sólido y versátil de la ingeniería del software.

2. Introducción

El software profesional, destinado a usarse por alguien más aparte de su desarrollador, se lleva a cabo en general por equipos, en vez de individualmente. Se mantiene y cambia a lo largo de su vida. La ingeniería de software busca apoyar el desarrollo de software profesional, en lugar de la programación individual. Incluye técnicas que apoyan la especificación, el diseño y la evolución del programa, ninguno de los cuales son normalmente relevantes para el desarrollo de software personal. Con el objetivo de ayudarlo a obtener una amplia visión de lo que trata la ingeniería de software. No obstante, cuando se habla de ingeniería de software, esto no sólo se refiere a los programas en sí, sino también a toda la documentación asociada y los datos de configuración requeridos para hacer que estos programas operen de manera correcta. [4]

3. Definiciones y conceptos

3.1 Antecedentes SWEBOK

3.1.1 Origen

La Computer Society comenzó a definir este cuerpo de conocimiento en 1998 como un paso necesario para hacer de la ingeniería de software una disciplina de ingeniería legítima y una profesión reconocida. Además, su primera versión fue publicada en el 2004.[1]
El proyecto está promovido por la IEEE Computer Society y la ACM (Association for Computing Machinery). El objetivo de este proyecto internacional es proporcionar una visión coherente de la ingeniería de software para los sectores público y privado.[9]

3.1.2 Estandarización

La estandarización del SWEBOK consiste en establecer un conjunto reconocido de normas y directrices que definen las áreas clave de conocimiento en ingeniería del software. Esto incluye:

- Definición de áreas de conocimiento.
- Define el contenido de la disciplina de ingeniería de software.
- Aclara los límites de la ingeniería de software en relación con otras disciplinas.
- Apoya la certificación y licencia de ingenieros de software.[10]

3.1.3 Colaboración internacional

El proyecto fue apoyado por un proceso de desarrollo en el que participaron aproximadamente 150 revisores de 33 países. [10]

3.2 Objetivo del SWEBOK

Los principales objetivos del proyecto SWEBOK incluyeron:

- Promover una visión coherente de la ingeniería de software en todo el mundo.
- Especificar el alcance y aclarar el lugar de la ingeniería de software con respecto a otras disciplinas como la informática, la gestión de proyectos, la ingeniería informática y las matemáticas.
- Caracterización de los contenidos de la disciplina de ingeniería de software.

- Proporcionar acceso tópico al Cuerpo de Conocimiento de Ingeniería de Software.
- Proporcionar una base para el desarrollo curricular y la certificación individual y material de licencia.[1]

3.3 Estructura

3.3.1 Áreas de conocimiento

El SWEBOK versión 3 se conforma de 15 áreas de conocimiento, dichas áreas son las siguientes:

1. Requisitos de Software: Un proyecto de software, simplificado al máximo, es básicamente la transformación de un conjunto de requisitos en un sistema informático. En consecuencia, si se parte de diferentes requisitos se obtendrán diferentes sistemas como resultado. He aquí uno de los componentes más importantes de la gestión de requisitos, puesto que si la descripción de los mismos no es adecuada o se desvía de lo que el cliente o los usuarios finales desean, entonces tal vez obtendremos un sistema perfecto (o tal vez no) pero es evidente que dicho sistema no satisfará las expectativas generadas.
   La obtención de requisitos es, en definitiva, un proceso muy complejo en el que intervienen diferentes personas, las cuales tienen distinta formación y conocimiento del sistema (desarrolladores, clientes, usuarios, etc.).[2]

2. Diseño de Software: Tomando como punto de partida los requisitos (funcionales y no funcionales), se pretende obtener una descripción de la mejor solución software/hardware que dé soporte a dichos requisitos, teniendo no solamente en cuenta aspectos técnicos, sino también aspectos de calidad, coste y plazos de desarrollo. Idealmente, se deberían plantear varios diseños alternativos que cumplan con los requisitos, para posteriormente hacer una elección de acuerdo a criterios de coste, esfuerzo de desarrollo o de calidad tales como la facilidad de mantenimiento. Es importante resaltar que en esta fase se pasa del qué (obtenido en la fase de requisitos) al cómo (que es el objetivo de la fase de diseño).[2]

3. Construcción de Software: Un conjunto de actividades que engloban fundamentalmente la codificación, pero también la verificación del código, su depuración y ciertos tipos de pruebas. Estas actividades parten de una especificación de requisitos detallados que fueron elaborados durante las actividades de diseño, y dan como resultado un software libre de errores que cumple con dicha especificación. No obstante, la comprobación exhaustiva de la corrección del software no puede hacerse en esta etapa, ya que necesita someterse a un proceso minucioso de pruebas que la determine.

Este proceso no forma parte de las actividades de construcción en sentido estricto.[2]

4. Pruebas de Software: Las pruebas de software son, en realidad, un elemento diferente dentro del proceso de desarrollo. Al contrario que el resto de actividades, su éxito radica en la detección de errores tanto en el propio proceso como en el software obtenido como resultado del mismo. Podría parecer extraño a primera vista el hecho de que encontrar errores en el software construido deba considerarse un éxito, pero la perspectiva del ingeniero de pruebas es distinta a la del resto de profesionales implicados en el desarrollo.
   Una prueba de software es todo proceso orientado a comprobar la calidad del software mediante la identificación de fallos en el mismo. La prueba implica necesariamente la ejecución del software.[2]

5. Mantenimiento de Software: Las actividades de mantenimiento son actividades de Ingeniería del Software orientadas a la modificación o cambio del mismo. Pero para introducir cambios, primero se necesita una comprensión del objeto que se ha de cambiar, y sólo después se podrá hacer efectiva la modificación requerida.
   Las actividades de mantenimiento suelen regirse por contratos de mantenimiento donde se especifican claramente las responsabilidades de cada parte (los desarrolladores y el cliente) en las actividades post-entrega. Es decir, en los contratos o planes de mantenimiento se especifica qué actividades se consideran dentro del contrato y cuáles no, y se delimita la forma y alcance de las solicitudes de actuación.[2]

6. Gestión de Configuración de Software: Es la disciplina que aplica dirección y control técnico y administrativo para: identificar y documentar las características físicas y funcionales de los elementos de configuración, controlar los cambios de esas características, registrar e informar del procesamiento de los cambios y el estado de la implementación, y verificar la conformidad con los requisitos especificados.[2]

7. Gestión de la Ingeniería de Software: La gestión de la ingeniería de software puede definirse como la aplicación de actividades de gestión (planificación, coordinación, medición, seguimiento, control y presentación de informes) para garantizar que los productos y servicios de ingeniería de software se entreguen de manera eficiente, eficaz y en beneficio de las partes interesadas.[1]

8. Proceso de Ingeniería de Software: Un proceso de ingeniería consiste en un conjunto de actividades interrelacionadas que transforman uno o más insumos en productos, al tiempo que consumen recursos para lograr la

transformación. Los procesos de ingeniería de software se ocupan de las actividades laborales que realizan los ingenieros de software para desarrollar, mantener y operar el software, como requisitos, diseño, construcción, pruebas, gestión de configuración y otros procesos de ingeniería de software.[10]

9. Modelos y Métodos de Ingeniería de Software: Los modelos y métodos de ingeniería de software imponen una estructura a la ingeniería de software con el objetivo de hacer que esa actividad sea sistemática, repetible y, en última instancia, más orientada al éxito. El uso de modelos proporciona un enfoque para la resolución de problemas, una notación y procedimientos para la construcción y el análisis de modelos. Los métodos proporcionan un enfoque para la especificación, el diseño, la construcción, la prueba y la verificación sistemática del software final y los productos de trabajo asociados.[1]

10. Calidad de Software: El software, como producto elaborado que es, se construye de acuerdo a procesos preestablecidos y controlados, en cuya producción se emplean “ingredientes humanos”, tecnológicos, etc. Por ello, la producción de software de calidad debe seguir una receta similar a la de cualquier otro producto: buenos ingredientes, construidos, ensamblados y verificados en un proceso de calidad. Sólo esto garantiza la calidad del producto final.
    En el desarrollo de un sistema de software, la calidad aparece por vez primera en los requisitos, que es donde se establecen los parámetros y criterios de calidad del software que se construirá. Las características de calidad que se definan en este momento serán la referencia de ahí en adelante, por lo que todo aquello que se establezca como requisito de calidad en este punto tendrá una enorme influencia, tanto en la forma en que posteriormente se medirá la calidad, como en los criterios utilizados para evaluar si los parámetros de calidad establecidos se cumplieron o no al final del desarrollo.[2]

11. Práctica Profesional de Ingeniería de Software: El área de conocimiento de Práctica Profesional de Ingeniería de Software se ocupa del conocimiento, las habilidades y las actitudes que los ingenieros de software deben poseer para practicar la ingeniería de software de manera profesional, responsable y ética. Debido a las aplicaciones generalizadas de los productos de software en la vida social y personal, la calidad de los productos de software puede tener un profundo impacto en nuestro bienestar personal y la armonía social.[1]

12. Economía de la Ingeniería de Software: La economía de la ingeniería de software trata de la toma de decisiones relacionadas con la ingeniería de software en un contexto empresarial. El éxito de un producto, servicio y solución de software depende de una buena gestión empresarial.[10]

13. Fundamentos de Computación: Abarca el entorno operativo y de desarrollo en el que el software evoluciona y se ejecuta. Debido a que ningún software puede existir en el vacío o ejecutarse sin una computadora, el núcleo de dicho entorno es la computadora y sus diversos componentes. El conocimiento sobre la computadora y sus principios subyacentes de hardware y software sirve como marco en el que se ancla la ingeniería de software.[1]

14. Fundamentos Matemáticos: Los profesionales del software viven con programas. En un lenguaje muy simple, uno puede programar sólo algo que siga una lógica bien entendida y no ambigua. El área de conocimiento Fundamentos matemáticos ayuda a los ingenieros de software a comprender esta lógica, que a su vez se traduce en código de lenguaje de programación. Las matemáticas que son el foco principal, son bastante diferentes de la aritmética típica, donde se manejan y discuten números. La lógica y el razonamiento son la esencia de las matemáticas que un ingeniero de software debe abordar.[1]

15. Fundamentos de Ingeniería: Describe algunas de las habilidades y técnicas fundamentales de la ingeniería que son útiles para un ingeniero de software. El enfoque se centra en temas que respaldan otras habilidades básicas y, al mismo tiempo, minimizan la duplicación de temas tratados con anterioridad.[10]

3.4 Evolución

El SWEBOK (Software Engineering Body of Knowledge) es un estándar que proporciona una guía comprensiva de los conocimientos y prácticas en la ingeniería de software. Desde su primera versión en 2004, ha evolucionado con varias versiones que reflejan los cambios en la disciplina y la industria.

- **SWEBOK v1.0 (2004)**: Estableció un marco inicial de conocimientos esenciales en ingeniería de software.
- **SWEBOK v2.0 (2004)**: Realizó mejoras y ajustes basados en la retroalimentación de la comunidad, consolidando áreas existentes y sugiriendo nuevas.
- **SWEBOK v3.0 (2014)**: Amplió las áreas de conocimiento, incorporando enfoques modernos como el desarrollo ágil y nuevas prácticas relevantes en la industria.
- **SWEBOK v4.0**: Actualmente en desarrollo, esta versión ampliará el contenido al incluir tres nuevas áreas de conocimiento, aumentando el total de 15 a 18. Las nuevas áreas serán:
    - Arquitectura de software.
    - Operaciones de ingeniería de software.

- Seguridad del software.

3.5 Actualización

En ingeniería, la acreditación de programas universitarios y la concesión de licencias y certificación de profesionales se consideran esenciales. Estos procesos son vitales para el desarrollo continuo de los profesionales y la mejora general de los estándares profesionales. Establecer un cuerpo central de conocimiento es fundamental para la creación y acreditación de planes de estudio universitarios, así como para la concesión de licencias y certificación de profesionales.

Alcanzar un consenso sobre este cuerpo central de conocimiento es un hito importante en cualquier disciplina. La IEEE Computer Society ha identificado esto como crucial para avanzar en la ingeniería de software hacia el reconocimiento profesional completo. La Guía SWEBOK, desarrollada bajo la Junta de Actividades Profesionales, es parte de un proyecto a largo plazo destinado a lograr este consenso. El proyecto SWEBOK en curso, con su próximo hito “SWEBOK Versión 4.0”, continuará redefiniendo “knowledge” aceptado e introduciendo nuevas áreas de enfoque.[1]

Ejemplos

A continuación, se presentan los 10 ejemplos que se utilizaron en el tutorial.

Pregunta 1.-

¿En qué año la Computer Society comenzó a definir este cuerpo de conocimiento y en que año fue publicada su primera versión?

Se comenzó a definir en el año:

- **1998**

Su primera versión fue publicada en el año:

- **2004**

Pregunta 2.-

¿Cuál es la institución que promueve el proyecto y cúal es el objetivo?

El proyecto está promovido por:

- **IEEE Computer Society y ACM (Association for Computing Machinery)**

Su objetivo es:

- **Proporcionar una visión coherente de la ingeniería de software para los sectores público y privado.**

Pregunta 3.-

¿En qué consiste la estandarización del SWEBOK?

La estandarización consiste en:

- **Establecer un conjunto de normas y directrices que definen las áreas clave de conocimiento de ingeniería de software.**

Pregunta 4.-

¿Cuáles son los 4 aspectos incluyen la estandarización del SWEBOK?

La estandarización incluye:

- **Definición de áreas de conocimiento**

Segundo punto

- **Define el contenido de la disciplina de ingeniería de software**

Tercer punto

- **Aclara los límites de la ingeniería de software en relación con otras disciplinas.**

Cuarto punto

- **Apoya la certificación y licencia de ingenieros de software.**

Pregunta 5.-

¿Cuántos revisores apoyaron en la elaboración del SWEBOK y cuántos países participan?

Participaron un total de:

- **150 revisores**

Los revisores que participaron fueron de:

- **33 países**

Pregunta 6.-

¿Cuáles son el primero, el tercero y el quinto objetivos del proyecto SWEBOK?

:
Primer objetivo:

- **Promover una visión coherente de la ingeniería de software en todo el mundo.**

Segundo objetivo:

- **Caracterización de los contenidos de la disciplina de ingeniería de software.**

Tercer objetivo:

- **Proporcionar una base para el desarrollo curricular y la certificación individual y material de licencia.**

Pregunta 7.-

¿Cuáles son las primeras cuatro áreas del conocimiento del SWEBOK?

:
Primer área:

- **Requisitos de Software.**

Segunda área:

- **Diseño de Software.**

Tercer área:

- **Construcción de Software.**

Cuarta área:

- **Pruebas de Software.**

Pregunta 8.-

¿Cuál es el área del conocimiento número 15 y en qué consiste?

:
El área número 15 del SWEBOK es:

- **Fundamentos de ingeniería.**

Consiste en:

- **Describe algunas de las habilidades y técnicas fundamentales de la ingeniería que son útiles para un ingeniero de software.**

Pregunta 9.-

¿En qué año fue lanzada la versión 1.0 y la versión 3.0 del SWEBOK ?

:
La versión 1.0 fue lanzada en::

- **2004**

La versión 3.0 fue lanzada en:

- **2014**

Pregunta 10.-

¿Cuáles son las tres nuevas áreas del conocimiento que va a tener SWEBOK versión 4.0?
:
La área número 16 será:

- **Arquitectura de software.**

La área número 17 será:

- **Operaciones de ingeniería de software.**

La área número 18 será:

- **Seguridad del software.**

Cuestionario

A continuación, se presentan 50 preguntas, las cuáles la respuesta en **negritas** es la respuesta correcta.

1. ¿Quiénes intervienen en la obtención de requisitos?

- **Desarrolladores, clientes, usuarios, etc.**
- Solo los desarrolladores.
- Exclusivamente los clientes.
- Únicamente los usuarios finales.

2. ¿Qué se toma como punto de partida en “Diseño de software”?

- **Requisitos funcionales y no funcionales**
- Las tecnologías disponibles en el mercado.
- La opinión de los inversores.
- Solo los requisitos de rendimiento.

3. ¿En diseño de software qué aspectos se toman en cuenta?

- **Técnicos, calidad, coste y plazos.**
- Exclusivamente la experiencia del usuario.
- Únicamente el coste y el tiempo de entrega.
- Solo los requerimientos funcionales.

4. ¿Qué actividades se engloban en la construcción de software?

- **Codificación, verificación de código, etc.**
- Solo la documentación del software.
- Exclusivamente el diseño de la interfaz.
- La planificación del proyecto, pero no la codificación.

5. ¿Qué comprueba la etapa de pruebas de software?

- **La calidad del software.**
- Solo la velocidad de desarrollo del software.
- La satisfacción del cliente, sin evaluar la calidad.
- Exclusivamente la documentación del proyecto.

6. ¿De quién se toman en cuenta los errores en la etapa de pruebas de software?

- **De todos los implicados en el desarrollo del software.**
- Solo de los desarrolladores.
- Exclusivamente de los usuarios finales.
- Únicamente de los testers.

7. ¿Por qué se dirige el mantenimiento?

- **Contratos.**
- La intuición del desarrollador..
- Normas de diseño gráfico.
- Los comentarios de los usuarios en redes sociales.

8. ¿Cuándo se hace el mantenimiento de software?

- **Post-entrega.**
- Durante la fase de diseño.
- Antes de la entrega final.
- Solo durante la fase de pruebas.

9. ¿Cuál es la disciplina que ayuda a identificar y documentar las características físicas y funcionales de los elementos de configuración?
- **Gestión de la configuración de software.**
- Desarrollo ágil de software.
- Análisis de requisitos.
- Pruebas de software.

10. ¿Qué se usa para construir un software de calidad?
- **Procesos preestablecidos y controlados**.
- La creatividad individual del programador.
- Herramientas de diseño gráfico.
- Solo el conocimiento técnico del equipo.

11. ¿Cuál es el objetivo principal del proyecto SWEBOK?
- Proporcionar herramientas de software gratuitas
- **Definir los límites y proporcionar acceso al conocimiento de la ingeniería de software**
- Crear nuevos lenguajes de programación
- Producir software para el sector público

12. ¿Qué organizaciones promueven el proyecto SWEBOK?
- Google y Microsoft
- **IEEE Computer Society y ACM**
- NASA y SpaceX
- Universidad de Stanford y MIT

13. ¿Cuántas áreas de conocimiento define SWEBOK para la ingeniería de software?
- 5
- 7
- **15**
- 11

14. ¿Cuál de las siguientes áreas NO es parte de las definidas en SWEBOK?
- Requisitos de software
- Construcción de software
- **Análisis de hardware**
- Pruebas de software

15. ¿En qué fase del proyecto se encuentra SWEBOK?
- Fase inicial
- **Fase de prueba**

- Fase final
- Fase de implementación

16. ¿Cómo se llama la tercera fase del proyecto SWEBOK?
- **Fase Iron Man**
- Fase Steel Man
- Fase Software Man
- Fase Power Man

17. ¿Qué contiene la versión de prueba 1.00 de SWEBOK?
- Solo referencias de investigación
- **Artículos científicos organizados en torno a áreas temáticas de ingeniería de software**
- Un manual de programación
- Herramientas de gestión de proyectos

18. ¿Cuál es el objetivo de los documentos en SWEBOK?
- Definir un modelo de negocio para el software
- **Definir terminología, conceptos y enfoques para áreas de conocimiento en ingeniería de software**
- Crear un nuevo lenguaje de programación
- Establecer estándares de hardware

19. ¿Qué sucederá con SWEBOK cuando finalice el proyecto?
- **Se convertirá en un estándar de la IEEE**
- Se lanzará como un producto comercial
- Será parte del estándar ISO 9000-3
- No tendrá cambios significativos

20. ¿Qué significa que SWEBOK aún no sea un estándar?
- No ha sido aceptado por la comunidad académica
- **Todavía está en fase de desarrollo y prueba**
- Ya es obsoleto
- Ha sido rechazado como estándar

21. ¿Cuál de los siguientes aspectos no es claramente abordado por SWEBOK según los criterios CSF?
- **CSF dinámicos**
- CSF estáticos
- CSF de programación
- CSF de hardware

22. ¿A quién está dirigida la guía SWEBOK?
- Solo a estudiantes de ingeniería de software
- **A organismos públicos, privados, académicos, y profesionales de ingeniería de software**
- Únicamente a programadores principiantes
- Exclusivamente a desarrolladores de software internacionales

23. ¿Cuál es uno de los objetivos de SWEBOK para los organismos públicos y privados?
- Enseñar lenguajes de programación avanzados
- **Proporcionar una visión estable de la ingeniería de software para definir requerimientos en educación y competencias laborales**
- Crear nuevas tecnologías de hardware
- Facilitar herramientas de software de código abierto

24. ¿Qué tipo de políticas y normas pretende beneficiar SWEBOK?
- Políticas de seguridad en la red
- **Normas de certificación, acreditación y guías para la práctica profesional**
- Regulaciones de privacidad de datos
- Normas de marketing digital

25. ¿Qué comité fue establecido para coordinar la ingeniería de software y promover la profesionalización de esta área?
- Comité de Tecnologías Digitales
- **Comité Coordinador de Ingeniería de Software (SWECC)**
- Comité de Desarrollo de Software Avanzado
- Comité de Ética en Programación

26. ¿En qué año fue iniciado el proyecto SWEBOK por SWECC?
- 1995
- **1998**
- 2000
- 2003

27. ¿Qué institución fue contratada para gestionar el proyecto SWEBOK?
- **Universidad de Quebec en Montreal (UQAM)**
- Universidad de California en Berkeley
- Universidad de Toronto
- Universidad de Harvard

28. ¿Cuántas fases tuvo el proyecto SWEBOK y cómo se llaman?
- Dos fases: Fase de Inicio y Fase de Ejecución
- **Tres fases: Hombre de Paja, Hombre de Piedra y Hombre de Hierro**
- Cuatro fases: Concepto, Desarrollo, Pruebas y Implementación
- Cinco fases: Investigación, Desarrollo, Prueba, Revisión y Publicación

29. ¿Qué fase del proyecto SWEBOK marca el inicio de la versión de prueba para uso?
- Fase de Inicio
- Hombre de Hierro
- **Hombre de Piedra**
- Hombre de Paja

30. ¿Cuál es la característica principal de la fase Hombre de Hierro en el proyecto SWEBOK?
- Es la fase inicial del proyecto
- **Es la fase en la que se alcanza un consenso mediante revisiones y pruebas**
- Es una fase en la que se lanzan prototipos de software
- Se enfoca en la implementación de políticas de seguridad

31. ¿Cuál fue el objetivo principal de la versión actual de la guía SWEBOK?
- Reducir el número de áreas de conocimiento
- **Mejorar la legibilidad, consistencia y usabilidad de la guía**
- Agregar nuevos temas sin aprobación de consenso
- Traducir el contenido a varios idiomas

32. ¿Qué se hizo para mejorar la consistencia en la guía SWEBOK?
- Se eliminaron varias áreas de conocimiento
- **Se reescribió todo el texto para unificar el estilo a lo largo del documento**
- Se aumentó la cantidad de contenido técnico avanzado
- Se simplificó el lenguaje para principiantes

33. ¿Cuál de las siguientes áreas de conocimiento (AC) fue considerada difícil de aplicar en un contexto práctico?
- Gestión del software
- **Construcción del software**
- Pruebas de software
- Configuración de software

34. ¿Qué problema se identificó en el área de conocimiento de gestión en la versión de prueba de SWEBOK?
- Era demasiado específica en la ingeniería de software
- No estaba alineada con los estándares de la industria
- **Estaba muy cercana al concepto de gestión en general**
- Mezclaba conceptos de calidad del producto y proceso

35. ¿Cuál fue la crítica principal al área de conocimiento de calidad en la guía SWEBOK?
- No incluía técnicas de calidad modernas
- **Mezclaba la calidad en el proceso y calidad en el producto**
- No cubría la calidad en el proceso
- No era fácil de entender

36. ¿Qué cambios se realizaron en algunas áreas de conocimiento después de la fase de prueba?
- Se fusionaron en una sola área de conocimiento
- **Se revisaron para eliminar material que ya estaba en otras áreas**
- Se actualizaron con nuevos temas sin aprobación
- Se tradujeron a otros idiomas

37. ¿Qué se menciona como una de las limitaciones de la guía SWEBOK?
- La guía se enfoca en técnicas obsoletas
- **La ingeniería de software evoluciona continuamente con nuevas tecnologías y prácticas**
- No incluye referencias a literatura científica
- Solo es aplicable en contextos académicos

38. ¿Por qué se seleccionaron las referencias en la guía SWEBOK?
- Por ser las únicas disponibles en el tema
- **Por estar en inglés, ser accesibles, recientes y fáciles de leer**
- Por ser referencias definitivas en el campo
- Por su antigüedad y relevancia histórica

39. ¿Cómo describe la guía SWEBOK la selección de referencias incluidas en ella?
- Como una selección exhaustiva y definitiva
- **Como una selección razonable, no definitiva**
- Como una recopilación limitada a artículos científicos
- Como un compendio de todas las referencias disponibles

40. ¿Qué tipo de referencias fueron omitidas en la guía SWEBOK debido a una limitación mencionada?
- Referencias que no son recientes
- **Referencias que no están en inglés**
- Referencias de autores no reconocidos
- Referencias que son difíciles de leer

41. ¿Qué recomendación se dio después del periodo de prueba de la guía SWEBOK?
- Revisar completamente todas las áreas de conocimiento
- **Reescribir tres áreas de conocimiento específicas**
- Reducir el número de áreas de conocimiento a la mitad
- Traducir la guía a otros idiomas

42. ¿Por qué es posible que SWEBOK necesite ser actualizado en el futuro?
- Porque la guía se quedó obsoleta en el momento de su publicación
- **Debido a la evolución continua de la ingeniería de software y la aparición de nuevas técnicas**
- Para cumplir con los estándares de hardware modernos
- Para incluir autores y referencias no seleccionadas

43. ¿Cómo define SWEBOK un requerimiento de software?
- Como una funcionalidad opcional del software
- **Como una propiedad que debe exhibir el software para resolver un problema del mundo real**
- Como un aspecto estético del diseño de la interfaz
- Como una parte secundaria del proceso de desarrollo

44. ¿Cuál es la primera subárea de conocimiento en el área de Requerimientos del Software según SWEBOK?
- Validación de Requerimientos
- **Fundamentos de los Requerimientos del Software**
- Captura de Requisitos
- Análisis de Requerimientos

45. ¿Qué tipos de requerimientos se mencionan en la subárea de Fundamentos de los Requerimientos del Software?
- Software vs. hardware, obligatorio vs. opcional
- **Producto vs. proceso, funcional vs. no funcional, propiedades emergentes**
- Estructural vs. funcional, interno vs. externo

- Diseño vs. desarrollo, orientado a usuarios vs. orientado a sistema

46. ¿Cuál es el propósito de la subárea de Validación de Requerimientos?
- Realizar pruebas de rendimiento del sistema
- Redactar el documento de especificación
- **Descubrir problemas en los requerimientos antes de asignar recursos**
- Documentar el diseño arquitectónico del software

47. ¿Qué actividades incluye la subárea de Análisis de Requerimientos?
- Redacción de manuales de usuario y diseño de interfaz
- **Clasificación, modelado conceptual, diseño arquitectural, asignación y negociación de requerimientos**
- Creación de prototipos y pruebas de aceptación
- Revisión de especificaciones y optimización de código

48. ¿Cuál de las siguientes opciones describe correctamente la subárea de Especificación de Requerimientos?
- Se enfoca en la elaboración de un único documento de requerimientos para todos los proyectos
- **Generalmente implica la producción de un documento que puede ser revisado y aprobado sistemáticamente**
- Su objetivo principal es la validación del sistema desarrollado
- Consiste únicamente en la captura de requerimientos funcionales

49. ¿Qué tema NO está incluido en la subárea de Consideraciones Prácticas sobre los requerimientos del software en SWEBOK?
- Gestión de cambios
- Medición de los requerimientos
- Proceso iterativo de los requerimientos
- **Diseño gráfico de la interfaz de usuario**

50. ¿Cuál es el objetivo de la subárea de Captura de Requisitos en SWEBOK?
- Documentar los requisitos en un formato estandarizado
- Validar que los requisitos sean correctos y aceptables para el usuario
- **Identificar de dónde provienen los requerimientos y cómo el ingeniero de software puede obtenerlos**
- Evaluar el impacto de los cambios en los requisitos durante el desarrollo

Cuestionario [Programa]

A continuación, se presentan 6 preguntas en el apartado del programa, las cuáles, la respuesta en **negritas** es la respuesta correcta.

51. ¿Cómo define el IEEE el diseño de software en IEEE 610.12-90?
- Como el proceso de prueba de software
- **Como el proceso de definir la arquitectura, componentes, interfaces y otras características de un sistema o componente**
- Como el proceso de gestión de requisitos
- Como la implementación de código en lenguajes de programación

52. ¿Cuál es el propósito de la subárea de Fundamentos del Diseño del Software en SWEBOK?
- **Proporcionar una base para entender el rol y el ámbito del diseño del software**
- Enseñar cómo implementar código en varios lenguajes de programación
- Describir los pasos para probar un sistema
- Proveer un marco de trabajo para la gestión de proyectos

53. ¿Qué tema NO está incluido en la subárea de Temas Clave en el Diseño del Software?
- Concurrencia
- Control y manejo de eventos
- **Diseño de interfaces de usuario**
- Distribución de componentes

54. ¿Qué se analiza en la subárea de Calidad del Diseño del Software en SWEBOK?
- **Atributos de calidad, análisis de calidad y técnicas de evaluación y medición**
- Estrategias para implementar algoritmos eficientes
- Técnicas para mejorar la experiencia del usuario
- Métodos de documentación y control de versiones

55. ¿En qué se dividen las Notaciones del Diseño del Software según SWEBOK?
- Estructuras arquitecturales y puntos de vista
- **Descripciones estructurales y de comportamiento**
- Métodos funcionales y basados en componentes
- Pruebas unitarias y de integración

56. ¿Qué estrategias y métodos se describen en la subárea de Estrategias y Métodos del Diseño del Software?
- Solo métodos funcionales y orientados a objetos
- **Métodos funcionales, orientados a objetos, centrados en la estructura de datos, basados en componentes y otros**
- Técnicas de modelado y pruebas de software
- Control de versiones y despliegue de software

Guión

[INTRODUCCIÓN]

Hola, hoy hablaremos sobre una de las áreas de conocimiento del SWEBOK: la gestión de configuración del software.

Primero, vamos a dar un breve contexto sobre el SWEBOK. Este término se refiere al Software Engineering Body of Knowledge o Cuerpo de Conocimientos en Ingeniería de Software. El SWEBOK es un estándar desarrollado para resumir las mejores prácticas, métodos y principios de la ingeniería de software. Está dividido en varias áreas de conocimiento, como:

- Requisitos de software
- Diseño de software
- Pruebas de software
- Mantenimiento

Y, por supuesto, la gestión de configuración del software

Cada área se centra en un aspecto clave del desarrollo de software. En particular, la gestión de configuración es fundamental para controlar y rastrear cambios en el software, asegurando que cada modificación sea registrada y gestionada correctamente a lo largo del ciclo de vida del proyecto.

Para entender mejor cómo funciona la gestión de configuración, vamos a verlo en acción utilizando Git, una herramienta popular de control de versiones que muchos equipos de desarrollo utilizan para este fin.

[DESARROLLO]

1. Identificación de Configuración

En la gestión de configuración, necesitamos identificar los elementos de software que se deben controlar. Estos elementos son todos aquellos artefactos que tienen un impacto en el software final, como:

Archivos de código fuente (main.py, index.html, login.py),

Documentación como el README.md,

Archivos de configuración necesarios para que el sistema funcione, como config.json o settings.xml.

Imaginemos que trabajamos en un proyecto llamado MiApp. En este proyecto, cada archivo se organizará en una estructura de carpetas específica y quedará registrado en el repositorio de Git. Con esta identificación clara, sabemos qué elementos están bajo control y cuáles no, lo que facilita el seguimiento y la organización a medida que el proyecto crece.

2. Control de Versiones

En esta fase, utilizamos un sistema como Git para mantener el control de cada modificación realizada. En Git, esto se gestiona mediante ramas, que nos permiten trabajar en diferentes partes del proyecto de manera independiente.

Tenemos una rama principal (main) para la versión estable del proyecto, y otras ramas para trabajar en diferentes funcionalidades o correcciones de errores.

Por ejemplo, si necesitamos agregar una nueva funcionalidad de "Login" a MiApp, crearíamos una nueva rama llamada feature/login para trabajar solo en esa parte del código. Esto nos permite realizar cambios y probar la funcionalidad sin afectar la versión principal.

Entonces, al trabajar en la rama feature/login, podemos hacer todas las modificaciones necesarias en el archivo login.py. Cada cambio se guarda en lo que llamamos un commit, y todos los commits de esta rama se mantienen separados de la rama principal hasta que estemos listos para integrarlos.

3. Seguimiento del Cambio

Cada cambio en el proyecto necesita ser registrado y documentado para mantener un historial detallado. Con Git, cada vez que hacemos un cambio en un archivo, guardamos ese cambio en un commit que incluye:

Un mensaje descriptivo sobre el cambio realizado,

La fecha,

Y el autor del cambio.

Supongamos que hemos agregado el formulario de inicio de sesión en login.py. Creamos un commit con el mensaje: “Añadido formulario de inicio de sesión en login.py”. De esta forma, si alguien revisa el historial, puede ver quién hizo este cambio, cuándo, y el propósito de la modificación. Este registro detallado es crucial para mantener la trazabilidad en el proyecto y permite regresar a versiones anteriores si es necesario.

4. Auditoría de la Configuración

Antes de integrar los cambios en la versión estable, es crucial hacer una revisión de código.

Esto se realiza mediante un proceso de pull request en Git.

Los desarrolladores pueden revisar los cambios propuestos, hacer sugerencias y asegurarse de que los nuevos cambios cumplen con los estándares.

Por ejemplo, antes de que la funcionalidad de "Login" sea parte de la rama principal, otros miembros del equipo revisarán el código en feature/login. Así, evitamos introducir errores en la versión estable y aseguramos que todo funcione como debe.

5. Generación de Reportes y Auditorías

Finalmente, la gestión de configuración permite generar reportes de cada versión.

En Git, esto incluye un registro detallado de los commits, el historial de versiones, y el estado actual del proyecto.

Estos reportes son útiles para realizar auditorías. Por ejemplo, si alguien pregunta qué cambios se hicieron en la última versión de MiApp, podemos revisar el historial de commits para ver cada modificación, desde la función de inicio de sesión hasta cualquier cambio menor.

[CONCLUSIÓN]

En resumen, podemos decir que la gestión de configuración del software es fundamental para mantener un desarrollo controlado, seguro y colaborativo. Con herramientas como Git, podemos gestionar versiones, rastrear cambios, realizar revisiones de código y generar reportes detallados para garantizar la integridad del software en todo momento.

Bibliografía

1. IEEE Computer Society. *Software Engineering Body of Knowledge (SWEBOK)*. https://www.computer.org/education/bodies-of-knowledge/software-engineering

2. Alonso, S., Sicilia Urban, M. A., & Rodríguez García, D. (2012). *Ingeniería de software: Un enfoque desde la guía SWEBOK*. Alfaomega Grupo Editor; Garceta Grupo Editorial.

3. Pressman, R. S. (2019). *Ingeniería de software: Un enfoque práctico* (7ª ed.). McGraw Hill.

4. Sommerville, I. (2011). *Ingeniería de software* (9ª ed.). Pearson Educación

5. Brooks, F. P. Jr. (1975). *The mythical man-month: Essays on software engineering*. Addison-Wesley.

6. Jones, C. (2010). *Software engineering best practices: Lessons from successful projects in the top companies*. McGraw-Hill.

7. Gorton, I. (2006). *Essential software architecture*. Springer-Verlag.

8. Galin, D. (2004). *Software quality assurance: From theory to implementation*. Pearson Education.

9. Komi-Sirviö, S. (2004). *Development and evaluation of software process improvement methods*. Vit.

10. IEEE Computer Society. (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), Version 3.0*. IEEE. https://www.computer.org/education/bodies-of-knowledge/software-engineering

Anexo

KA	Área de conocimiento
LIBRO SUECO	Cuerpo de conocimientos de ingeniería de software

La publicación de la versión 2004 de esta Guía del conjunto de conocimientos de ingeniería de software (SWE-BOK 2004) fue un hito importante en el establecimiento de la ingeniería de software como una disciplina de ingeniería reconocida. El objetivo de desarrollar esta actualización de SWEBOK es mejorar la actualidad, la legibilidad, la coherencia y la facilidad de uso de la Guía.

Todas las áreas de conocimiento (KAS) se han actualizado para reflejar los cambios en la ingeniería de software desde la publicación de SWEBOK 2004. Se han añadido cuatro nuevas KA fundamentales y una KA de prácticas profesionales de ingeniería de software. La KA de herramientas y métodos de ingeniería de software se ha revisado como Modelos y métodos de ingeniería de software. Las herramientas de ingeniería de software son ahora un tema en cada una de las KA. Tres apéndices proporcionan las especificaciones para la descripción de la KA, un conjunto anotado de estándares relevantes para cada KA y una lista de las referencias citadas en la Guía.

Esta guía, escrita bajo los auspicios del Consejo de Actividades Profesionales de la IEEE Computer Society, representa un siguiente paso en la evolución de la profesión de ingeniería de software.

¿QUÉ ES LA INGENIERÍA DE SOFTWARE?

El Vocabulario de Ingeniería de Software y Sistemas ISO/IEC/IEEE (SEVOCAB) define la ingeniería de software como "la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería al software").

¿CUALES SON LOS OBJETIVOS DE LA GUÍA SWEBOK?

La Guía no debe confundirse con el Cuerpo de Conocimiento en sí, que existe en forma publicada.

1 Consulte www.computer.org/sevocab.

Literatura. El propósito de la Guía es describir la parte del Cuerpo de Conocimiento que es generalmente aceptada, organizar esa parte y proporcionar acceso temático a ella.

La Guía del Cuerpo de Conocimientos de Ingeniería de Software (Guía SWEBOK) se creó con los siguientes cinco objetivos:

1. Promover una visión coherente de la ingeniería de software en todo el mundo.
2. Especificar el alcance y aclarar el lugar de la ingeniería de software con respecto a otras disciplinas como la informática, la ingeniería de proyectos, etc. Gestión de proyectos, ingeniería informática y matemáticas.
3. Caracterizar los contenidos de la disciplina de ingeniería de software.
4. Proporcionar un acceso temático al Cuerpo de Conocimiento de Ingeniería de Software
5. Proporcionar una base para el desarrollo del plan de estudios y para el material de certificación y licencia individual.

El primero de estos objetivos, una visión mundial coherente de la ingeniería de software, fue apoyado por un proceso de desarrollo en el que participaron aproximadamente 150 revisores de 33 países. Puede encontrarse más información sobre el proceso de desarrollo en el sitio web (www.swebok.org). Se estableció contacto con sociedades científicas y profesionales y agencias públicas involucradas en la ingeniería de software, se les informó sobre este proyecto para actualizar SWEBOK y se les invitó a participar en el proceso de revisión. Se reclutaron editores de KA de América del Norte, la Cuenca del Pacífico y Europa. Se realizaron presentaciones sobre el proyecto en varios lugares internacionales.

El segundo de los objetivos, el deseo de especificar el alcance de la ingeniería de software, motiva la organización fundamental de la Guía. El material que se reconoce como perteneciente a esta disciplina se organiza en las quince áreas de conocimiento enumeradas en la Tabla I.1. Cada una de estas áreas de conocimiento se trata en un capítulo de esta Guía.

xxii

Tabla 1.1. Los 15 KAS de SWEBOK
Requerimientos del software
Diseño del software
Construcción del software
Pruebas del software
Mantenimiento del software
Gestión de la configuración del software
Gestión de la ingeniería del software
Proceso de ingeniería del software
Modelos y métodos de ingeniería del software
Calidad del software
Práctica profesional de la ingeniería del software
Economía de la ingeniería del software
Fundamentos de la informática
Fundamentos matemáticos
Fundamentos de la ingeniería

Al especificar el alcance, también es importante identificar las disciplinas que se cruzan con la ingeniería del software. Para este fin, SWEBOK V3 también reconoce siete disciplinas relacionadas, que se enumeran en la Tabla 1.2. Los ingenieros de software deben, por supuesto, tener conocimiento del material de estas disciplinas (y las descripciones de KA en esta Guía pueden hacer referencia a ellas). Sin embargo, no es un objetivo de la Guía SWEBOK caracterizar el conocimiento de las disciplinas relacionadas.

Tabla 1.2. Disciplinas relacionadas
Ingeniería informática
Ciencias de la computación
Gestión general
Matemáticas
Gestión de proyectos
Gestión de la calidad
Ingeniería de sistemas

Los elementos relevantes de la informática y las matemáticas se presentan en la Fundamentos de la informática y Fundamentos matemáticos KAS de la Guía (Capítulos 13 y 14).

ORGANIZACIÓN JERÁRQUICA

La organización de los capítulos de KA apoya el tercero de los objetivos del proyecto: una caracterización de los contenidos de la ingeniería de software. Las especificaciones detalladas proporcionadas por el equipo editorial del proyecto a los editores asociados con respecto al contenido de las descripciones de las KA se pueden encontrar en el Apéndice A.

La Guía utiliza una organización jerárquica para descomponer cada KA en un conjunto de temas con etiquetas reconocibles. Un desglose de dos (a veces tres) niveles proporciona una manera razonable de encontrar temas de interés. La Guía trata los temas seleccionados de una manera compatible con las principales escuelas de pensamiento y con los desgloses que se encuentran generalmente en la industria y en la literatura y los estándares de ingeniería de software. Los desgloses de los temas no presuponen dominios de aplicación particulares, usos comerciales, filosofías de gestión, métodos de desarrollo, etc. La extensión de la descripción de cada tema es solo la necesaria para comprender la naturaleza generalmente aceptada de los temas y para que el lector encuentre contexto al material de referencia; el Cuerpo de conocimientos se encuentra en los materiales de referencia en sí, no en la Guía.

MATRIZ Y MATERIAL DE REFERENCIA

Para proporcionar acceso temático al conocimiento, el cuarto objetivo del proyecto, la Guía identifica material de referencia autorizado para cada KA. El Apéndice C proporciona una Lista de referencia consolidada para la Guía. Cada KA incluye referencias relevantes de la Lista de referencia consolidada y también incluye una matriz que relaciona el material de referencia con los temas incluidos.

Cabe señalar que la Guía no intenta ser exhaustiva en sus citas. Gran parte del material que es adecuado y excelente no está referenciado. El material incluido en la Lista de referencia consolidada proporciona cobertura de los temas descritos.

PROFUNDIDAD DEL TRATAMIENTO

Para lograr el quinto objetivo de SWEBOK: proporcionar una base para el desarrollo curricular,

CAPÍTULO 7

GESTIÓN DE INGENIERÍA DE SOFTWARE

ACRÓNIMOS

Guía del PMBOK	Guía para la gestión de proyectos Cuerpo de conocimientos
SDLC	Ciclo de vida del desarrollo de software
CUAL	Gestión de ingeniería de software
———	Garantía de calidad del software
SWX	Extensión de software al PMBOK® Guía
EDT	Estructura de desglose del trabajo

INTRODUCCIÓN

La gestión de la ingeniería de software puede definirse como la aplicación de actividades de gestión (planificación, coordinación, medición, seguimiento, control y presentación de informes) para garantizar que los productos y servicios de ingeniería de software se entreguen de manera eficiente, eficaz y en beneficio de las partes interesadas. La disciplina relacionada con la gestión es un elemento importante de todas las áreas de conocimiento (AC), pero, por supuesto, es más relevante para esta AC que para otras AC. La medición también es un aspecto importante de todas las AC; el tema de los programas de medición se presenta en esta AC.

En cierto sentido, debería ser posible gestionar un proyecto de ingeniería de software de la misma manera que se gestionan otros proyectos complejos. Sin embargo, existen aspectos específicos de los proyectos de software y de los procesos del ciclo de vida del software que complican una gestión eficaz, entre ellos:

1 Los términos Iniciación, Planificación, Ejecución, Seguimiento y Control, y Cierre se utilizan para describir los grupos de procesos en la Guía del PMBOK y SWX.

- Los clientes a menudo no saben qué se necesita o qué es factible.
- Los clientes a menudo no aprecian las complejidades inherentes a la ingeniería de software, particularmente con respecto al impacto de los requisitos cambiantes.
- Es probable que una mayor comprensión y condiciones cambiantes generen requisitos de software nuevos o modificados.
- Como resultado de los requisitos cambiantes, el software a menudo se construye utilizando un proceso iterativo en lugar de una secuencia de tareas cerradas.
- La ingeniería de software necesariamente incorpora creatividad y disciplina. Mantener un equilibrio adecuado entre ambas es a veces difícil.
- El grado de novedad y complejidad suele ser alto.
- A menudo se produce un rápido ritmo de cambio en la tecnología subyacente.

Las actividades de gestión de la ingeniería de software se dan en tres niveles: gestión de la organización y de la infraestructura, gestión de proyectos y gestión del programa de medición. Los dos últimos se tratan en detalle en esta descripción de la KA. Sin embargo, esto no pretende restar importancia a las cuestiones de gestión de la organización y de la infraestructura. En general, se acepta que los gerentes de ingeniería organizacional de software deben estar familiarizados con los conocimientos de gestión de proyectos y medición de software descritos en esta KA. También deben poseer algún conocimiento del dominio objetivo. Asimismo, también es útil que los gerentes de proyectos y programas complejos en los que el software es un componente de la arquitectura del sistema sean conscientes de las diferencias que los procesos de software introducen en la gestión de proyectos y la medición de proyectos.

CAPÍTULO 8

PROCESO DE INGENIERÍA DE SOFTWARE

ACRÓNIMOS

BPMN	Notación de modelado de procesos de negocio
CASO	Ingeniería de software asistida por computadora
—	Gestión de configuración
CMMI	Modelo de madurez de capacidades Integración
GQM	Objetivo-Pregunta-Métrica
IDEF	Definición de integración
LEER	Nivel de esfuerzo
ODC	Clasificación de defectos ortogonales
SDLC	Ciclo de vida del desarrollo de software
SPLC	Ciclo de vida del producto de software
UML	Lenguaje de modelado unificado

INTRODUCCIÓN

Un proceso de ingeniería consiste en un conjunto de actividades interrelacionadas que transforman uno o más insumos en productos, al tiempo que consumen recursos para lograr la transformación. Muchos de los procesos de las disciplinas de ingeniería tradicionales (por ejemplo, eléctrica, mecánica, civil, química) se ocupan de transformar la energía y las entidades físicas de una forma a otra, como en una represa hidroeléctrica que transforma la energía potencial en energía eléctrica o una refinería de petróleo que utiliza procesos químicos para transformar el petróleo crudo en gasolina.

En esta área de conocimiento (KA), los procesos de ingeniería de software se ocupan de las actividades laborales que realizan los ingenieros de software para desarrollar, mantener y operar el software, como requisitos, diseño, construcción, pruebas, gestión de configuración y otros procesos de ingeniería de software. Para facilitar la lectura, "ingeniería de software"

En este KA, al "proceso de software" se le denominará "proceso de software". Además, tenga en cuenta que "proceso de software" denota actividades de trabajo, no el proceso de ejecución del software implementado.

Los procesos de software se especifican por varias razones: para facilitar la comprensión, la comunicación y la coordinación humanas; para ayudar a la gestión de proyectos de software; para medir y mejorar la calidad de los productos de software de manera eficiente; para apoyar la mejora de procesos; y para proporcionar una base para el soporte automatizado de la ejecución de procesos.

Los temas de SWEBOK relacionados estrechamente con este tema de KA de proceso de ingeniería de software incluyen la gestión de la ingeniería de software, los modelos y métodos de ingeniería de software y la calidad del software; el tema de medición y análisis de causa raíz que se encuentra en el tema de KA de Fundamentos de ingeniería también está estrechamente relacionado. La gestión de la ingeniería de software se ocupa de adaptar e implementar procesos de software para un proyecto de software específico (consulte la planificación de procesos en el tema de KA de gestión de la ingeniería de software). Los modelos y métodos respaldan un enfoque sistemático para el desarrollo y la modificación de software.

El KA de Calidad de Software se ocupa de los procesos de planificación, aseguramiento y control de la calidad de proyectos y productos. La medición y los resultados de la medición en el KA de Fundamentos de ingeniería son esenciales para evaluar y controlar los procesos de software.

DESGLOSE DE TEMAS PARA EL PROCESO DE INGENIERÍA DE SOFTWARE

Como se ilustra en la Figura 8.1, este KA se ocupa de la definición del proceso de software, la vida útil del software ciclos, evaluación y mejora de procesos de software ment, medición de software e ingeniería de software Implementación de procesos de ingeniería

CAPÍTULO 9

MODELOS Y MÉTODOS DE INGENIERÍA DE SOFTWARE

ACRÓNIMOS

3GL	Lenguaje de tercera generación
BNF	Forma Backus-Naur
FDD	Desarrollo basado en características
IA	Desarrollo Integrado Ambiente
PBI	Elemento del backlog del producto
RAD	Desarrollo rápido de aplicaciones
UML	Lenguaje de modelado unificado
XP	Programación extrema

INTRODUCCIÓN

Los modelos y métodos de ingeniería de software imponen una estructura a la ingeniería de software con el objetivo de hacer que esa actividad sea sistemática, repetible y, en última instancia, más orientada al éxito. El uso de modelos proporciona un enfoque para la resolución de problemas, una notación y procedimientos para la construcción y el análisis de modelos. Los métodos proporcionan un enfoque para la especificación, el diseño, la construcción, la prueba y la verificación sistemática del software final y los productos de trabajo asociados.

Los modelos y métodos de ingeniería de software varían ampliamente en su alcance, desde abordar una sola fase del ciclo de vida del software hasta cubrir el ciclo de vida completo del software. El énfasis en esta área de conocimiento (CA) está en los modelos y métodos de ingeniería de software que abarcan múltiples fases del ciclo de vida del software, ya que los métodos específicos para fases individuales del ciclo de vida están cubiertos por otras CA.

DESGLOSE DE TEMAS PARA MODELOS Y MÉTODOS DE INGENIERÍA DE SOFTWARE

Este capítulo sobre modelos y métodos de ingeniería de software se divide en cuatro áreas temáticas principales:

- Modelado. Analiza la práctica general del modelado y presenta temas sobre principios de modelado; propiedades y expresión de modelos; sintaxis, semántica y pragmática del modelado; y precondiciones, poscondiciones e invariantes.
- Tipos de modelos: analiza brevemente los modelos y la agregación de submodelos y proporciona algunas características generales de los tipos de modelos que se encuentran comúnmente en la práctica de la ingeniería de software.
- Análisis de modelos: presenta algunos de los técnicas de análisis comunes utilizadas en el modelado para verificar la integridad, la consistencia, la corrección, la trazabilidad y la interacción. Métodos de ingeniería de software presenta un breve resumen de los métodos de ingeniería de software comúnmente utilizados. La discusión guía al lector a través de un resumen de métodos heurísticos, métodos formales, creación de prototipos y métodos ágiles.

El desglose de los temas para el curso Modelos y métodos de ingeniería de software KA se muestra en la Figura 9.1.

1. Modelado

El modelado de software se está convirtiendo en una técnica generalizada para ayudar a los ingenieros de software a comprender

CAPÍTULO 11

PRÁCTICA PROFESIONAL DE INGENIERÍA DE SOFTWARE

ACRÓNIMOS

ACM	Asociación de Maquinaria Computacional
BCS	Sociedad Británica de Computación
CSDA	Asociado certificado en desarrollo de software
PCSD	Profesional certificado en desarrollo de software
CEI	Comisión Electrotécnica Internacional
IEEECS	Sociedad de Computación IEEE
IFIP	Federación Internacional para el Procesamiento de la Información
IP	Propiedad Intelectual
ISO	Organización Internacional de Normalización
ANS	Acuerdo de confidencialidad
OMPI	Organización Mundial de la Propiedad Intelectual
OMC	Organización Mundial del Comercio

INTRODUCCIÓN

El área de conocimiento de Práctica Profesional de Ingeniería de Software (KA) se ocupa del conocimiento, las habilidades y las actitudes que los ingenieros de software deben poseer para practicar la ingeniería de software de manera profesional, responsable y ética. Debido a las aplicaciones generalizadas de los productos de software en la vida social y personal, la calidad de los productos de software puede tener un profundo impacto en nuestro bienestar personal y la armonía social. Los ingenieros de software deben manejar problemas de ingeniería únicos, produciendo software con características y confiabilidad conocidas. Este requisito exige ingenieros de software que posean un conjunto adecuado de conocimientos, habilidades, capacitación y experiencia en la práctica profesional.

El término "práctica profesional" se refiere a una La práctica profesional es una forma de llevar a cabo servicios de manera que se alcancen determinados estándares o criterios tanto en el proceso de prestación de un servicio como en el producto final resultante del servicio. Estos estándares y criterios pueden incluir aspectos tanto técnicos como no técnicos. El concepto de práctica profesional puede considerarse más aplicable en aquellas profesiones que cuentan con un conjunto de conocimientos generalmente aceptados, códigos de ética y conducta profesional con sanciones por infracciones, procesos aceptados de acreditación, certificación y licencias, y sociedades profesionales que proporcionan y administran todos estos elementos. La admisión a estas sociedades profesionales suele basarse en una combinación prescrita de educación y experiencia.

Un ingeniero de software mantiene una práctica profesional al realizar todo el trabajo de acuerdo con las prácticas, estándares y pautas generalmente aceptadas, en particular las establecidas por la sociedad profesional correspondiente. Por ejemplo, la Association for Computing Machinery (ACM) y la IEEE Computer Society (IEEE CS) han establecido un Código de ética y práctica profesional de ingeniería de software. Tanto la British Computer Society (BCS) como la International Federation for Information Processing (IFIP) han establecido estándares de práctica profesional similares. ISO/IEC e IEEE han proporcionado además estándares de ingeniería de software aceptados internacionalmente (consulte el Apéndice B de esta Guía). IEEE CS ha establecido dos programas de certificación internacionales (CSDA, CSDP) y una Guía correspondiente al Cuerpo de conocimientos de ingeniería de software (Guía SWEBOK). Todos estos son

CAPÍTULO 12

ECONOMÍA DE LA INGENIERÍA DE SOFTWARE

ACRÓNIMOS

EVM	Gestión del valor ganado
TIR	Tasa interna de retorno
MARR	Tasa de retorno mínima aceptable
SDLC	Ciclo de vida del desarrollo de software
SPLC	Ciclo de vida del producto de software
———	Retorno de la inversión
AÑOS	Retorno sobre el capital empleado
———	Costo total de propiedad

INTRODUCCIÓN

La economía de la ingeniería de software trata de la toma de decisiones relacionadas con la ingeniería de software en un contexto empresarial. El éxito de un producto, servicio y solución de software depende de una buena gestión empresarial. Sin embargo, en muchas empresas y organizaciones, las relaciones comerciales del software con el desarrollo y la ingeniería de software siguen siendo vagas. Esta área de conocimiento (CA) proporciona una descripción general de la economía de la ingeniería de software.

La economía es el estudio del valor, los costos, los recursos y su relación en un contexto o situación determinados. En la disciplina de la ingeniería de software, las actividades tienen costos, pero el software resultante también tiene atributos económicos. La economía de la ingeniería de software proporciona una forma de estudiar los atributos del software y de los procesos de software de una manera sistemática que los relaciona con medidas económicas. Estas medidas económicas se pueden sopesar y analizar al tomar decisiones que están dentro del alcance de una organización de software y aquellas dentro del alcance integrado de un negocio completo de producción o adquisición.

La economía de la ingeniería de software se ocupa de alinear las decisiones técnicas del software con los objetivos comerciales de la organización. En todos los tipos de organizaciones (ya sean "con fines de lucro", "sin fines de lucro" o gubernamentales), esto se traduce en mantenerse en el negocio de manera sostenible. En las organizaciones "con fines de lucro", esto se relaciona además con lograr un rendimiento tangible del capital invertido, tanto de los activos como del capital empleado. Este KA se ha formulado de manera que se adapte a todos los tipos de organizaciones, independientemente de su enfoque, cartera de productos y servicios, o propiedad del capital y restricciones impositivas.

Decisiones como "¿Deberíamos utilizar un componente específico?" pueden parecer fáciles desde una perspectiva técnica, pero pueden tener implicaciones serias en la viabilidad comercial de un proyecto de software y el producto resultante. A menudo, los ingenieros se preguntan si tales preocupaciones se aplican en absoluto, ya que son "solo ingeniería". El análisis económico y la toma de decisiones son consideraciones de ingeniería importantes porque los ingenieros son capaces de evaluar las decisiones tanto técnicamente como desde una perspectiva comercial. Los contenidos de esta área de conocimiento son temas importantes que los ingenieros de software deben conocer incluso si nunca están involucrados en decisiones comerciales concretas; tendrán una visión completa de los problemas comerciales y el papel que juegan las consideraciones técnicas en la toma de decisiones comerciales. Muchas propuestas y decisiones de ingeniería, como fabricar versus comprar, tienen profundos impactos económicos intrínsecos que deben considerarse explícitamente.

Este KA cubre primero los fundamentos, la terminología clave, los conceptos básicos y las prácticas comunes de la economía de la ingeniería de software para indicar cómo la toma de decisiones en la ingeniería de software incluye, o debería incluir, una perspectiva empresarial. Luego proporciona una perspectiva del ciclo de vida, destaca la gestión de riesgos e incertidumbre y muestra cómo se utilizan los métodos de análisis económico. Algunas consideraciones prácticas finalizan el conocimiento. Área del tutorial.

CAPÍTULO 13

FUNDAMENTOS DE LA COMPUTACIÓN

ACRÓNIMOS

POA	Programación Orientada a Aspectos
IR	Unidad de Aritmética y Lógica
API	Interfaz de programación de aplicaciones
———	Modo de transferencia asincrónica
B/S	Navegador-Servidor
———	Equipo de respuesta ante emergencias informáticas
COTS	Productos comerciales listos para usar
CRUD	Crear, leer, actualizar, eliminar
C/S	Cliente-Servidor
CS	Ciencias de la Computación
SGBD	Sistema de gestión de bases de datos
FPU	Unidad de punto flotante
E/S	Entrada y salida
ISA	Arquitectura del conjunto de instrucciones
ISO	Organización Internacional de Normalización
ISP	Proveedor de servicios de Internet
LAN	Red de área local
MUX	Multiplexor
NADA	Tarjeta de interfaz de red
ORIENTADO	Programación orientada a objetos
OS	Sistema operativo
OSI	Interconexión de sistemas abiertos
———	Ordenador personal
PDA	Asistente digital personal
P2P	Protocolo punto a punto
RFID	Identificación por radiofrecuencia
RAM	Memoria de acceso aleatorio
———	Memoria de solo lectura

SCSI	Interfaz de sistema informático pequeño
SQL	Lenguaje de consulta estructurado
TCP	Protocolo de control de transporte
UDP	Protocolo de datagramas de usuario
VPN	Red privada virtual
WAN	Red de área amplia

INTRODUCCIÓN

El alcance del área de conocimiento de Fundamentos de la Computación (KA) abarca el entorno operativo y de desarrollo en el que el software evoluciona y se ejecuta. Debido a que ningún software puede existir en el vacío o ejecutarse sin una computadora, el núcleo de dicho entorno es la computadora y sus diversos componentes. El conocimiento sobre la computadora y sus principios subyacentes de hardware y software sirve como marco en el que se anida la ingeniería de software. Por lo tanto, todos los ingenieros de software deben tener una buena comprensión de — de la informática KA Se acepta generalmente que la ingeniería de software se basa en la ciencia de la computación. Por ejemplo, "Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering" [1] afirma claramente: "Un aspecto particularmente importante es que la ingeniería de software se basa en la ciencia de la computación y las matemáticas" (cursiva añadida).

Steve Tockey escribió en su libro Return on Software:

> Tanto la informática como la ingeniería de software se ocupan de las computadoras, la computación y el software. La ciencia de la computación, como cuerpo de conocimientos, es el núcleo de ambas.

CAPÍTULO 14

FUNDAMENTOS MATEMÁTICOS

INTRODUCCIÓN

Los profesionales del software viven con programas. En un lenguaje muy simple, uno puede programar sólo algo que siga una lógica bien entendida y no ambigua.

El área de conocimiento Fundamentos matemáticos (KA) ayuda a los ingenieros de software a comprender esta lógica, que a su vez se traduce en código de lenguaje de programación. Las matemáticas que son el foco principal de este KA son bastante diferentes de la aritmética típica, donde se manejan y discuten números. La lógica y el razonamiento son la esencia de las matemáticas que un ingeniero de software debe abordar.

Las matemáticas, en cierto sentido, son el estudio de los sistemas formales. La palabra "formal" se asocia con la precisión, por lo que no puede haber ninguna interpretación ambigua o errónea del hecho. Las matemáticas son, por tanto, el estudio de todas y cada una de las verdades ciertas sobre cualquier concepto. Este concepto puede referirse tanto a números como a símbolos, imágenes, sonidos, videos... casi cualquier cosa. En resumen, no sólo los números y las ecuaciones numéricas están sujetos a la precisión. Por el contrario, un ingeniero de software necesita tener una abstracción precisa sobre un dominio de aplicación diverso.

El KA Fundamentos matemáticos de la Guía SWEBOK cubre técnicas básicas para identificar un conjunto de reglas para el razonamiento en el contexto del sistema en estudio. Todo lo que se pueda deducir siguiendo estas reglas es una certeza absoluta dentro del contexto de ese sistema.

En este KA, se definen y discuten técnicas que pueden representar y llevar adelante el razonamiento y el juicio de un ingeniero de software de una manera precisa (y por lo tanto matemática). El lenguaje y los métodos de lógica que se discuten aquí nos permiten describir pruebas matemáticas para inferir de manera concluyente la verdad absoluta de ciertos conceptos más allá de los números.

En resumen, puedes escribir un programa para un problema solo si sigue cierta lógica. El objetivo de este KA es ayudarte a desarrollar la habilidad de identificar y describir dicha lógica. El énfasis está puesto en ayudarte a comprender los conceptos básicos en lugar de desafiar tus habilidades aritméticas.

DESGLOSE DE TEMAS PARA FUNDAMENTOS MATEMÁTICOS

El desglose de los temas para el KA de Fundamentos Matemáticos se muestra en la Figura 14.1.

1. Conjunto, Relaciones, Funciones

[1*, c2]

Set: Un conjunto es una colección de objetos, llamados elementos del conjunto. Un conjunto se puede representar enumerando sus elementos entre llaves, p. ej., S = {1, 2, 3}.

El símbolo ∈ se utiliza para expresar que un elemento pertenece a un conjunto, o en otras palabras es miembro del conjunto. Su negación se representa por ∉. Por ejemplo, 1 ∈ S, pero 4 ∉ S. En una representación más compacta de un conjunto utilizando la notación de construcción de conjuntos, {x | P(x)} es el conjunto de todos los x tales que P(x) para cualquier proposición P(x) sobre cualquier universo de discurso. Algunos ejemplos de conjuntos importantes incluyen los siguientes:

- N = {0, 1, 2, 3, ...} = el conjunto de números enteros no negativos.
- Z = {..., -3, -2, -1, 0, 1, 2, 3, ...} = el conjunto de números enteros.

Conjunto finito e infinito. Un conjunto con un número finito de elementos se llama conjunto finito. Por el contrario, cualquier conjunto que no tenga un número finito de elementos es un conjunto infinito. El conjunto de todos los números naturales, por ejemplo, es un conjunto infinito.

CAPÍTULO 15

FUNDAMENTOS DE INGENIERÍA

ACRÓNIMOS

CAIVALLA	Diseño asistido por computadora
CMMI	Modelo de madurez de capacidades Integración
pdf	Función de densidad de probabilidad
pmf	Función de masa de probabilidad
RCA	Análisis de causa raíz
SDLC	Ciclo de vida del desarrollo de software

INTRODUCCIÓN

El IEEE define la ingeniería como "la aplicación de un enfoque sistemático, disciplinado y cuantificable a estructuras, máquinas, productos, sistemas o procesos" [1].

Este capítulo describe algunas de las habilidades y técnicas fundamentales de la ingeniería que son útiles para un ingeniero de software. El enfoque se centra en temas que respaldan otras habilidades básicas y, al mismo tiempo, minimizan la duplicación de temas tratados en otras partes de este documento.

A medida que la teoría y la práctica de la ingeniería de software maduran, es cada vez más evidente que la ingeniería de software es una disciplina de ingeniería que se basa en conocimientos y habilidades comunes a todas las disciplinas de ingeniería. Esta área de conocimiento de Fundamentos de Ingeniería (KA) se ocupa de los fundamentos de ingeniería que se aplican a la ingeniería de software y otras disciplinas de ingeniería. Los temas de esta KA incluyen métodos empíricos y técnicas experimentales; análisis estadístico; medición; diseño de ingeniería; modelado, creación de prototipos y simulación; estándares; y análisis de causa raíz. La aplicación de este conocimiento, según corresponda, permitirá a los ingenieros de software desarrollar y mantener software de manera más eficiente y eficaz. Completar su trabajo de ingeniería de manera eficiente y eficaz.

La eficacia es un objetivo de todos los ingenieros en todas las disciplinas de ingeniería.

DESGLOSE DE TEMAS PARA FUNDAMENTOS DE INGENIERÍA

El desglose de los temas para el curso KA de Fundamentos de ingeniería se muestra en la Figura 15.1.

1. Métodos empíricos y técnicas experimentales

[2*, c1]

Un método de ingeniería para la resolución de problemas implica proponer soluciones o modelos de soluciones y luego realizar experimentos o pruebas para estudiar las soluciones o modelos propuestos. Por lo tanto, los ingenieros deben comprender cómo crear un experimento y luego analizar los resultados del experimento para evaluar la solución propuesta. Los métodos empíricos y las técnicas experimentales ayudan al ingeniero a describir y comprender la variabilidad en sus observaciones, a identificar las fuentes de variabilidad y a tomar decisiones.

Los tres tipos diferentes de estudios empíricos que se utilizan habitualmente en los proyectos de ingeniería son los experimentos diseñados, los estudios observacionales y los estudios retrospectivos. A continuación se ofrecen breves descripciones de los métodos más utilizados.

1.1. Experimento diseñado

Un experimento diseñado o controlado es una investigación de una hipótesis comprobable en la que se manipulan una o más variables independientes para medir su efecto sobre una o más variables dependientes. Una condición previa para realizar un experimento es la existencia de una hipótesis clara. Es importante que un ingeniero comprenda cómo formular hipótesis claras.

4.3 Introducción

Dentro de las actividades de la ingeniería del software, las relacionadas con la obtención y gestión de los requisitos resultan especialmente críticas. Este conjunto de actividades consiste en obtener y documentar los requisitos para desarrollar el sistema y posteriormente analizarlos con el objetivo de responder a la pregunta «¿Qué debe hacerse?».

Un proyecto software, simplificado al máximo, es básicamente la transformación de un conjunto de requisitos en un sistema informático. En consecuencia, si se parte de diferentes requisitos se obtendrán diferentes sistemas como resultado. He aquí uno de los componentes más importantes de la gestión de requisitos, puesto que si la descripción de los mismos no es adecuada, o se desvía de lo que el cliente o los usuarios finales desean, entonces tal vez obtendremos un sistema perfecto (o tal vez no) pero es evidente que dicho sistema no satisfará las expectativas generadas. Diremos entonces que el proyecto en cuestión ha fracasado (*véase* la Figura 4.1). Establecer con exactitud los requisitos de un sistema es, por tanto, un principio esencial para llevar a cabo con éxito un desarrollo de software. Como veremos, se trata de una de las más complicadas tareas a que se enfrentan los desarrolladores.

Figura 4.1: Las dificultades de comunicación en la descripción de los requisitos.

La obtención de requisitos es, en definitiva, un proceso muy complejo en el que intervienen diferentes personas, las cuales tienen distinta formación y conocimiento del sistema (desarrolladores, clientes, usuarios, etc.) En dicho proceso, es necesario tener en cuenta la influencia de diversos factores, que a veces de forma no evidente, tienen un impacto en el desarrollo del proceso en sí:

- En primer lugar hay que tener en cuenta la complejidad del problema a resolver. Como se apuntó en el primer capítulo, la complejidad es un factor inherente al software, y uno de los motivos fundamentales por los que ésto es así radica en el hecho de que con software se resuelven, generalmente, problemas complejos. La determinación exacta de los requisitos de un sistema que se adivina complejo no es, en la mayoría de los casos, tarea sencilla.

5.3 Introducción

En la fase de diseño, tomando como punto de partida los requisitos (funcionales y no funcionales), se pretende obtener una descripción de la *mejor* solución software/hardware que dé soporte a dichos requisitos, teniendo no solamente en cuenta aspectos técnicos, sino también aspectos de calidad, coste y plazos de desarrollo. Idealmente, se deberían plantear varios diseños alternativos que cumplan con los requisitos, para posteriormente hacer una elección de acuerdo a criterios de coste, esfuerzo de desarrollo o de calidad tales como la facilidad de mantenimiento. Es importante resaltar que en esta fase se pasa del *qué* (obtenido en la fase de requisitos) al *cómo* (que es el objetivo de la fase de diseño).

Veamos un ejemplo muy sencillo de lo anterior, plasmado en un sistema de facturación. En este sistema, el qué (un requisito funcional), podría ser que las facturas se impriman ordenadas por código postal para que la empresa se beneficie de la rebaja que por ello hace la oficina de correos. En cuanto al cómo, habría que pensar en el diseño del módulo de ordenación y la comunicación con el proceso general de facturación, y a más bajo nivel, en la decisión del algoritmo de ordenación más adecuado de entre los que cumplan los requisitos funcionales y no funcionales.

Además de todo ello, en la fase de diseño se pasa a un ámbito más técnico donde se habla fundamentalmente el lenguaje de los desarrolladores ya que los productos del diseño son documentos y artefactos orientados a ingenieros del software, y poco o nada el de los clientes, pues la comunicación con los clientes o usuarios es limitada, si bien puede ser necesario generar documentación de diseño para los clientes llegado el caso. Con todo lo anterior en mente, es momento de definir formalmente el diseño. Concretamente, el Glosario IEEE de Términos de Ingeniería del Software (IEEE, 1990) lo define indistintamente con las dos acepciones siguientes:

> El **diseño** puede definirse como (1) el proceso para definir la arquitectura, los componentes, los interfaces, y otras características de un sistema o un componente, y (2) como el resultado de este proceso

En esta definición se mencionan varios elementos importantes en los diseños software como por ejemplo la arquitectura, que tiene que ver –como estudiaremos más adelante– con la descomposición de un sistema en sus componentes e interfaces. No obstante, no aclara un hecho importante sobre la estructura del propio proceso de diseño, y que resulta esencial introducir ya. Nos referimos a que el diseño se descompone claramente en dos subprocesos que son los siguientes:

- Diseño de la arquitectura o de alto nivel, en el cual se describe cómo descomponer el sistema y organizarlo en los diferentes componentes (lo que se conoce como «la arquitectura del software»).

6.2 Objetivos

El propósito fundamental de este capítulo es estudiar los principios en los que debe basarse la construcción de software de calidad, describiendo algunas de las técnicas de Ingeniería del Software que deben tenerse en cuenta cuando se construyen programas.

Los objetivos específicos son los siguientes:

- Reconocer la importancia de la construcción de software como parte esencial en el proceso de desarrollo.
- Comprender el concepto de construcción como la creación de software en un lenguaje de programación, junto con las microtareas de diseño y pruebas asociadas.
- Estudiar los principios que gobiernan la construcción de software de calidad.
- Conocer técnicas de implementación que resultan de utilidad desde el punto de vista de la Ingeniería del Software.
- Comprender el papel de la reutilización en la construcción de software, conociendo sus beneficios y teniendo en cuenta sus riesgos.

6.3 Introducción

El término *construcción de software* define un conjunto de actividades que engloban fundamentalmente la codificación, pero también la verificación del código, su depuración y ciertos tipos de pruebas. Estas actividades parten de una especificación de requisitos detallados que fueron elaborados durante las actividades de diseño, y dan como resultado un software libre de errores que cumple con dicha especificación. No obstante, la comprobación exhaustiva de la corrección del software no puede hacerse en esta etapa, ya que necesita someterse a un proceso minucioso de pruebas que la determine. Este proceso no forma parte de las actividades de construcción en sentido estricto.

El producto resultante de la construcción de software, el software propiamente dicho, es en último término el entregable más importante de un proyecto de desarrollo. Ciertamente es el motivo principal por el que se está llevando a cabo. Sin embargo, no todo el mundo tiene la misma concepción de la construcción de software. En los modelos de ciclo de vida más tradicionales, este término es sinónimo de codificación. Estos enfoques estudian la construcción como una actividad claramente separada del diseño y las pruebas, que consiste en implementar, utilizando un cierto lenguaje de programación, las soluciones elaboradas durante el diseño y plasmadas en diversos diagramas, teniendo en cuenta las restricciones y la documentación generada durante dicha etapa. Otros modelos más modernos, particularmente los métodos ágiles –que se caracterizan por iterar sobre cortos ciclos de desarrollo– y los métodos basados en la evolución de prototipos, tienen un concepto distinto de la construcción, pues incluyen como parte de la misma la codificación, pero también

en la obtención de un nivel adecuado de calidad, y que las actividades de prueba son sus máximos garantes. Como se verá más adelante, el riesgo de prescindir de las pruebas, o de realizarlas de manera inadecuada, resulta tan elevado que no es posible hablar de un proceso de Ingeniería del Software sin ellas.

Las pruebas de software son en realidad un elemento diferente dentro del proceso de desarrollo. Al contrario que el resto de actividades, su éxito radica en la detección de errores tanto en el propio proceso como en el software obtenido como resultado del mismo. Podría parecer extraño a primera vista el hecho de que encontrar errores en el software construido deba considerarse un éxito, pero la perspectiva del ingeniero de pruebas es distinta a la del resto de profesionales implicados en el desarrollo. Sin embargo, es en cierto modo similar a lo que ocurre en otras áreas, como por ejemplo las pruebas diagnósticas en el campo de la Medicina. Una prueba diagnóstica se considera positiva si detecta algo mal, si bien, es mucho peor para un paciente tener algo mal y no saberlo o tener conocimiento de que algo está mal sin conocer con exactitud qué.

> Una **prueba de software** es todo proceso orientado a comprobar la calidad del software mediante la identificación de fallos en el mismo. La prueba implica necesariamente la ejecución del software

Dada su especial naturaleza, debe tenerse muy en cuenta que el éxito de las pruebas depende, en buena medida, de la actitud de colaboración que todo el equipo de desarrollo muestre respecto a las actividades de pruebas como garantía de calidad del software que se está produciendo. A este respecto, algunos autores abogan por fomentar un ambiente favorable respecto al descubrimiento de fallos, evitando en lo posible que nadie se sienta culpable por los errores que se vayan descubriendo.

Durante las pruebas se tiene presente el hecho de que los usuarios finales potencialmente ejecutarán todas las funcionalidades del sistema, lo que implica que el software será puesto a prueba inevitablemente, bien por los desarrolladores o bien por los usuarios finales. Por tanto, los errores que no sean detectados en las pruebas realizadas durante el desarrollo aparecerán cuando los usuarios finales utilicen el software, con el consiguiente impacto en la imagen y reputación del equipo de desarrollo, esfuerzo y coste asociado a la reparación, posibles implicaciones legales a consecuencia del mal funcionamiento del software, etc. Debe intentarse, y éste es uno de los objetivos fundamentales de las pruebas, que el porcentaje de fallos detectados por el usuario sea lo menor posible.

Probar un producto no es sino comparar su estado actual con el estado esperado de acuerdo a una especificación del mismo. Por extensión, y puesto que el software no es otra cosa que el producto resultante de un proceso de desarrollo, todo proceso que permite comprobar el comportamiento de un determinado software con el comportamiento que se espera del mismo según lo descrito en sus especificaciones podría ser catalogado como prueba de software, siendo los aspectos fundamentales a verificar en un software su corrección, com-

8.3 Introducción

Comenzaremos la discusión de este capítulo recordando la definición posiblemente más utilizada de la Ingeniería del Software. Esta definición, que se citó en el capítulo de introducción, decía:

> La Ingeniería del Software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, la operación y el **mantenimiento** del software; esto es, la aplicación de la ingeniería al software

En la propia definición aparece el mantenimiento como una de las fases de la Ingeniería del Software. Ahora bien ¿en qué se diferencia el mantenimiento de otras actividades de la Ingeniería del Software? –más concretamente, ¿en qué se diferencia de lo que se suele denominar desarrollo? Para clarificar esta delimitación, hay que buscar un criterio que separe unas actividades de las otras, como se intentará a lo largo de este capítulo.

En una primera aproximación podemos decir que las actividades de mantenimiento son actividades de Ingeniería del Software orientadas a la *modificación o cambio* del mismo. Pero para introducir cambios, primero se necesita una comprensión del objeto que se ha de cambiar, y sólo después se podrá hacer efectiva la modificación requerida. Esto hace que la comprensión del software –como actividad humana– sea un elemento esencial en la Ingeniería del Software y obliga a que este último tenga una cierta estructura e incluya una documentación asociada que facilite su comprensión.

La importancia del mantenimiento radica en sus implicaciones económicas, al igual que sucede en otras actividades de ingeniería. En un sistema fácil de mantener se puede implementar un cambio con un menor esfuerzo, lo que permite pensar que el trabajo de hacer el software más fácil de mantener puede resultar rentable. Claro está, esto sólo será cierto si se cumple la premisa de que los cambios son frecuentes en el software. Dado que la experiencia nos demuestra que todo software evoluciona para adaptarse a las necesidades de sus usuarios, la premisa anterior se cumple para la mayor parte de los casos. En consecuencia, prever el mantenimiento del software y trabajar –aun antes de la entrega– con el objetivo de hacerlo más fácil de mantener, aunque ello en un principio aumente los costes de producción, merece la pena pues estos sobrecostes se recuperarán con creces si el mantenimiento resulta más fácil.

En la siguiente sección se definen y comentan algunos conceptos fundamentales que se manejarán en el resto del capítulo, importantes para comprender el estado actual de la cuestión y conseguir una comprensión de las actividades relacionadas con el mantenimiento.

8.4 Conceptos fundamentales

Antes de profundizar en los temas que se tratan en este capítulo, es preciso definir formalmente conceptos fundamentales como *mantenimiento* o *facilidad de mantenimiento*, detallar aquellos aspectos que influyen en la facilidad de mantenimiento, analizar sus propiedades y comprender el papel que otorgan al mantenimiento de software algunos modelos de calidad ampliamente aceptados.

8.4.1 ¿Qué es el mantenimiento del software?

La consideración de qué tipo de actividades de Ingeniería del Software se consideran mantenimiento ha evolucionado con el tiempo. Es importante conocer las diferentes concepciones del término, pues las actividades de mantenimiento suelen regirse por contratos de mantenimiento donde se especifican claramente las responsabilidades de cada parte (los desarrolladores y el cliente) en las actividades post-entrega. Es decir, en los contratos o planes de mantenimiento se especifica qué actividades se consideran dentro del contrato y cuáles no, y se delimita la forma y alcance de las solicitudes de actuación. Todo lo cual necesita un consenso previo sobre lo que se considera mantenimiento y lo que no.

Cualquier esfuerzo de Ingeniería del Software –si termina con éxito– acaba por producir un determinado producto software, orientado a satisfacer ciertos requisitos previamente establecidos. El mantenimiento en este contexto se entiende de manera general como las actividades que producirán cambios en el producto inicialmente acordado. Ahora bien, el cambio se puede entender de diferentes maneras. El estándar 1219 de IEEE para el mantenimiento del software (IEEE, 1998c) define mantenimiento como:

> El **mantenimiento** del software es la modificación de un producto software después de la entrega para corregir fallos, para mejorar su rendimiento u otros atributos, o para adaptar el producto a un entorno modificado

Esta definición considera el mantenimiento una actividad post-entrega, situando las actividades de mantenimiento de un producto sólo después de que dicho producto haya sido entregado y puesto en operación.

Otras fuentes consideran que algunas actividades de mantenimiento pueden comenzar antes de la entrega del producto, como ocurre, por ejemplo, cuando se revisa el diseño antes de la implementación para mejorar la facilidad de mantenimiento futura. Algunas de estas actividades son la planificación de las actividades posteriores a la entrega o las acciones orientadas a facilitar el mantenimiento posterior, como por ejemplo la revisión de la documentación. Si bien se trata, en cierto modo, de actividades de preparación para el mantenimiento más que de mantenimiento en sí, su inclusión aporta una visión más amplia del mantenimiento del software. En esta línea, Pigoski resalta la necesidad de comenzar a considerar el mantenimiento desde el mismo momento en que comienza el desarrollo:

De este modo, se puede considerar la configuración de un software como una *sucesión de configuraciones en el tiempo*. Por lo tanto, hay dos maneras de ver la configuración: en un instante concreto del tiempo, como se considera en la primera acepción, o a través del tiempo, según la historia de los cambios. Lógicamente, no es necesario en todas las ocasiones registrar cada uno de los cambios que se producen en los elementos que conforman el software, sino sólo en aquellos que se consideren relevantes o definitivos.

De las definiciones anteriores se obtiene la siguiente (IEEE, 1990):

> La **gestión de la configuración** es la disciplina que aplica dirección y control técnico y administrativo para: identificar y documentar las características físicas y funcionales de los elementos de configuración, controlar los cambios de esas características, registrar e informar del procesamiento de los cambios y el estado de la implementación, y verificar la conformidad con los requisitos especificados

siendo el elemento de configuración la unidad en gestión de la configuración.

> Un **elemento de configuración** es un agregado de hardware, software o ambos, al que se da una designación y se trata como una entidad independiente en el proceso de gestión de la configuración

Ahora bien, si restringimos los elementos de configuración solamente a aquellos que se consideran parte del software, hablaremos de *gestión de configuración del software*, cuyos elementos de configuración llamaremos *elementos de configuración software* o *elementos software*. De ello se deduce que la gestión de la configuración del software consiste en un conjunto de actividades de *soporte* a otras actividades de Ingeniería del Software, pues todas las actividades –desde las de obtención de requisitos hasta las pruebas– generan artefactos (de muy diversa índole), que pueden considerarse elementos software. Teniendo en cuenta el valor de las actividades de gestión de la configuración del software como actividades de soporte, se puede dar una definición alternativa de gestión de la configuración:

> La **gestión de la configuración** también puede definirse como el control de las diferencias en el sistema para minimizar el riesgo y el error

A continuación, pasamos a examinar las actividades de la gestión de la configuración del software, junto con conceptos adicionales que son importantes para el ingeniero implicado en estas actividades.

El software, como producto elaborado que es, se construye de acuerdo a procesos preestablecidos y controlados, en cuya producción se emplean «ingredientes humanos», tecnológicos, etc. Por ello, la producción de software de calidad debe seguir una receta similar a la de cualquier otro producto: buenos ingredientes, construidos, ensamblados y verificados en un proceso de calidad. Sólo esto garantiza la calidad del producto final.

La calidad es un término del que todos tenemos una noción clara, y sin embargo nos resulta difícil definirla con palabras. Tiene además muchos aspectos, pues es un concepto aplicable a prácticamente cualquier objeto, proceso o acción. Así, nos referimos a una tela de calidad, a una enseñanza de calidad, e incluso, a una jugada de fútbol de gran calidad individual. Existen agencias dedicadas a velar por la calidad, a certificarla o a acreditar que un proceso o producto cumple unos ciertos parámetros de calidad pero... ¿qué es en realidad la calidad? La definición de Raymond Paul es una referencia importante a la hora de estudiar este concepto desde la perspectiva de la Ingeniería del Software: *«la característica que distingue el grado de excelencia o superioridad de un proceso, producto o servicio»*.

Una forma sencilla de evaluar la calidad de un producto es identificar aquellos atributos que diferencian a un objeto de calidad de uno que no la tiene. Si el producto en cuestión posee dichos atributos, entonces se considera que tiene calidad y en caso contrario, se considera que no la tiene. Así, cuando hablamos del software, la calidad se identifica a menudo con la ausencia de fallos. Sin embargo, esta definición es bastante laxa, puesto que hay fallos que estamos dispuestos a tolerar, por ejemplo, en un navegador web, pero que consideraríamos inaceptables en un software que controlase procesos críticos tales como el sistema de control de una aeronave comercial. Éste es en realidad el enfoque de algunos de los modelos de calidad del software que estudiaremos en el capítulo.

A lo largo de las diferentes secciones analizaremos cómo se entiende el concepto de calidad desde la perspectiva de la Ingeniería del Software. Como veremos, éste no se limita a la noción comentada de calificativo aplicable al producto resultante de un desarrollo, sino también al propio proceso de ingeniería y a la organización que lo lleva a cabo.

9.2 Objetivos

El objetivo general de este capítulo es presentar el concepto de calidad y analizar su importancia en el desarrollo de software. Así, tras la lectura de este capítulo el lector debería:

- Conocer los conceptos fundamentales de calidad según la perspectiva de la Ingeniería del Software.
- Conocer los diferentes modelos de calidad del software.
- Saber qué es un modelo de evaluación y mejora de procesos y los desafíos que su implantación plantea.
- Conocer los estándares vigentes sobre calidad del software, así como también otros modelos y especificaciones de actualidad relacionados con la misma.

9.3 Introducción

En la entrada de este capítulo hemos reflexionado sobre el concepto de calidad en general, como característica deseable, como calificativo de un producto, de un proceso o de una acción. Durante esa discusión se hizo referencia a los atributos de calidad de un producto software. Ahora que vamos a comenzar su estudio desde la perspectiva de la Ingeniería del Software, es necesario comenzar reconociendo los múltiples aspectos con los que está relacionado el término *calidad*. La guía SWEBOK indica expresamente que los ingenieros del software deben conocer el significado y características de la calidad, así como su valor en el desarrollo y mantenimiento del software.

En el desarrollo de un sistema de software, la calidad aparece por vez primera en los requisitos, que es donde se establecen los parámetros y criterios de calidad del software que se construirá. Las características de calidad que se definan en este momento serán la referencia de ahí en adelante, por lo que todo aquello que se establezca como requisito de calidad en este punto tendrá una enorme influencia, tanto en la forma en que posteriormente se medirá la calidad, como en los criterios utilizados para evaluar si los parámetros de calidad establecidos se cumplieron o no al final del desarrollo. La Figura 9.1 muestra los diferentes aspectos de la calidad desde el punto de vista de la Ingeniería del Sofwtare. A continuación estudiaremos cada uno de ellos con detalle.

Figura 9.1: Los múltiples aspectos de la calidad en la Ingeniería del Software

9.3.1 Cultura y ética de la calidad

En su libro «*Creación de una cultura de la Ingeniería del Software*», Karl Wiegers hace un detallado catálogo de aquellos elementos que contribuyen al éxito de un proyecto de software, poniendo un énfasis especial en la mejora de la cultura de la calidad en una organización. Esta cultura de la Ingeniería del Software se podría definir como el compromiso de los ingenieros del software de una organización con las metas de calidad de su organización y particularmente, con aquellas que tienen que ver con la obtención de software de calidad.

2.2.2.2 SWEBOK

El proyecto internacional SWEBOK (http://www.swebok.org/) actualmente en curso tiene como objetivo "ofrecer una caracterización validada por consenso de los límites de la disciplina de ingeniería de software y proporcionar un acceso temático al conjunto de conocimientos que sustenta esa disciplina" (SWEBOK 2001, p. i). El proyecto está promovido por la IEEE Computer Society y la ACM. El objetivo de este proyecto internacional es proporcionar una visión coherente de la ingeniería de software para los sectores público y privado. SWEBOK define nueve áreas de conocimiento para la ingeniería de software:

- Requisitos de software
- Construcción de software
- Pruebas de software
- mantenimiento de software
- Gestión de configuración de software
- Gestión de ingeniería de software
- Proceso de ingeniería de software
- Herramientas y métodos de ingeniería de software
- Calidad del software.

Todas estas áreas y sus subáreas se describen en detalle. El proyecto se encuentra actualmente en una fase de prueba y la versión final de SWEBOK debería estar disponible a finales del año 2003, cuando está previsto que finalice la tercera fase del proyecto denominada "Iron Man Phase".

2.4.1 El cuerpo de conocimientos de la ingeniería del software - SWEBOK

El cuerpo de conocimientos de una disciplina está constituido por toda su literatura. Para facilitar el manejo y acceso a este conocimiento, las organizaciones profesionales de dicha disciplina crean las guías o manuales que seleccionan, organizan y documentan la literatura más influyente que conforma dicho cuerpo de conocimientos.

La guía del cuerpo de conocimientos de la ingeniería del *software*, elaborada por la Sociedad de Computación de la IEEE y mejor conocida como Guía SWEBOK (IEEE CS, 2014), describe el conocimiento que es generalmente aceptado y que caracteriza a la ingeniería del *software*. Uno de sus usos más importantes está en la elaboración de programas curriculares y de certificación profesional en ingeniería del *software*.

La guía SWEBOK persigue tres fines relacionados:

1. Identificar aquellas partes del cuerpo de conocimientos sobre las cuales existe un consenso generalizado entre los miembros de la comunidad académica y profesional de la ingeniería del *software*.
2. Organizar estas partes en áreas de conocimiento.
3. Crear los medios necesarios para acceder a ellas.

Para cumplir con estos objetivos la IEEE CS ofrece esta guía en varios formatos, dos de ellos se distribuyen de manera gratuita y están disponibles en la dirección www.swebok.org. La versión 3.0 de la guía SWEBOK organiza el conocimiento de la ingeniería del *software* en quince secciones denominadas áreas de conocimiento. Estas áreas son las siguientes:

- Requisitos de *software*.
- Diseño de *software*.
- Construcción de *software*.
- Pruebas de *software*.
- Mantenimiento de *software*.
- Gestión de la configuración del *software*.
- Gestión de la ingeniería del *software*.
- Procesos de ingeniería del *software*.
- Modelos y métodos de ingeniería del *software*.
- Calidad del *software*.
- Práctica profesional de la ingeniería del *software*.
- Economía de la ingeniería del *software*.
- Fundamentos de computación.
- Fundamentos matemáticos.
- Fundamentos de ingeniería.

Para cada una de estas áreas la guía describe los conceptos fundamentales del área e indica cuál es su literatura más importante. Es, por lo tanto, un recurso educativo de suprema importancia en la formación de los ingenieros de *software*.

La Guía SWEBOK tiene múltiples propósitos:

- Define el contenido de la disciplina de ingeniería de software;
- Promueve una comprensión consistente de la ingeniería de software en todo el mundo;
- Aclara los límites de la ingeniería de software en relación con otras disciplinas;
- Proporciona una base para materiales de capacitación y desarrollo curricular;
- Apoya la certificación y licencia de ingenieros de software.
- Refleja las prácticas actuales e integra tecnologías emergentes.

Objetivos Básicos para SWEBOK V4.0

Los principales objetivos del proyecto SWEBOK incluyeron:

- Promover una visión coherente de la ingeniería de software en todo el mundo;
- Especificar el alcance y aclarar el lugar de la ingeniería de software con respecto a otras disciplinas como la informática, la gestión de proyectos, la ingeniería informática y las matemáticas;
- Caracterización de los contenidos de la disciplina de ingeniería de software;
- Proporcionar acceso tópico al Cuerpo de Conocimiento de Ingeniería de Software;
- Proporcionar una base para el desarrollo curricular y la certificación individual y material de licencia

Introducción: Sommerville, I. (2011). *Ingeniería de software* (9ª ed.). Pearson Educación.

1.1 Desarrollo de software profesional

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

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

Muchos suponen que el software es tan sólo otra palabra para los programas de cómputo. No obstante, cuando se habla de ingeniería de software, esto no sólo se refiere a los programas en sí, sino también a toda la documentación asociada y los datos de configuración requeridos para hacer que estos programas operen de manera correcta. Un sistema