Universidad Nacional Autónoma de México

Facultad de Ingeniería.

Ingeniería de Software

Grupo: 04

Profesor: Ing. Orlando Zaldívar Zamorategui Equipo: 1

Tutorial IS 2025-2

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

Integrantes:

Del Toro Ortiz Juan Pablo Yasbeth

González Armas Dylan Arturo

Marquez Martinez Fatima Athziri

Vázquez Flores José Ángel

Ciudad Universitaria, CDMX, a 12/06/2025

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.
- Permitir al alumno realizar una evaluación didáctica para reforzar el aprendizaje del SWEBOK, sus áreas de conocimiento y su utilidad en como herramienta de consulta para el desarrollo profesional

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.

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.

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 4 se conforma de 18 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 estos 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 las actividades, su éxito radica en la detección de errores tanto en el propio proceso como en el software obtenido como resultado de este. 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 de este. 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 con 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

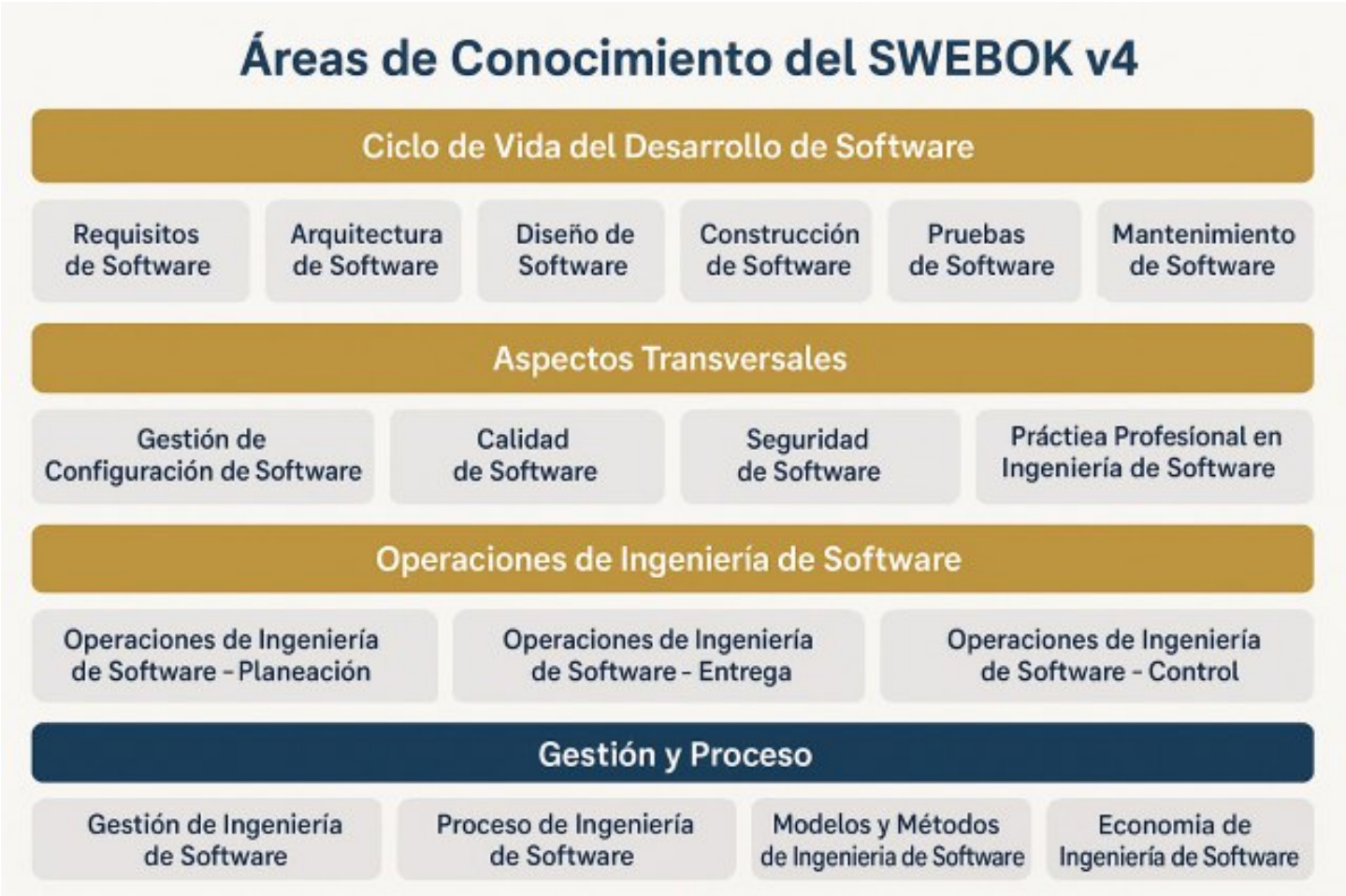
debe abordar.1

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

16. Fundamentos de informática: Cubre los conceptos y teorías fundamentales que sustentan la disciplina de la ingeniería de software.

17. Fundamentos matemáticos: Introduce técnicas y principios matemáticos relevantes para la ingeniería de software, como la lógica, la probabilidad y las matemáticas discretas.

18. Fundamentos de ingeniería: Cubre los principios y prácticas de ingeniería aplicables a la ingeniería de software, incluida la ingeniería de sistemas y la gestión de proyectos.



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: SWEBOK V4.0 (2024) Introduce nuevos temas, actualiza los existentes y elimina temas obsoletos. Agile y DevOps se han integrado notablemente en varias áreas de conocimiento debido a su amplia adopción desde la edición anterior. Además, se han añadido tres nuevas áreas de conocimiento: Arquitectura de Software, Operaciones de Ingeniería de Software y Seguridad de Software, para fortalecer los conocimientos básicos de la ingeniería de software.
- Por lo tanto, las áreas que contempla el SWEBOK más reciente comparado con su versión anterior son las siguientes

V3 (2014)	V4 (2024)
Introduction	Introduction
1. Software Requirements	1. Software Requirements
	2. Software Architecture
2. Software Design	3. Software Design
3. Software Construction	4. Software Construction
4. Software Testing	5. Software Testing
	6. Software Engineering Operations
5. Software Maintenance	7. Software Maintenance
6. Software Configuration Management	8. Software Configuration Management
7. Software Engineering Management	9. Software Engineering Management
8. Software Engineering Process	10. Software Engineering Process
9. Software Engineering Models and Methods	11. Software Engineering Models and Methods
10. Software Quality	12. Software Quality
	13. Software Security
11. Software Engineering Professional Practice	14. Software Engineering Professional Practice
12. Software Engineering Economics	15. Software Engineering Economics
13. Computing Foundations	16. Computing Foundations
14. Mathematical Foundations	17. Mathematical Foundations
15. Engineering Foundations	18. Engineering Foundations
Appendix A. Knowledge Area Specifications	Appendix A. Knowledge Area Specifications
Appendix B. Standards	Appendix B. Standards
Appendix C. Consolidated Reference List	Appendix C. Consolidated Reference List

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]

3.6 ¿Por qué SWEBOK es útil para los desarrolladores de software?

Los desarrolladores de software pueden utilizar SWEBOK para evaluar sus habilidades y carencias en diversos temas de ingeniería de software y encontrar recursos para aprender más sobre los principios y conocimientos de esta.

SWEBOK también puede ayudar a los desarrolladores a gestionar el progreso de sus competencias al ofrecer una estructura y un lenguaje común para las ideas y los métodos del campo.

3.7 ¿Cómo pueden los desarrolladores de software utilizar SWEBOK?

El SWEBOK puede ser empleado por desarrolladores como una guía para los siguientes aspectos:

- Comprobar sus propios conocimientos y experiencia en cada AC (Área de Conocimiento).
- Obtener más información de las referencias y enlaces de cada AC, así como de los temas relacionados y emergentes.
- Como una guía para comprender y utilizar términos, conceptos, mejores prácticas y estándares de ingeniería de software.
- Para planificar su desarrollo profesional, estableciendo objetivos de competencia basados en los temas.

3.8 Público Objetivo del SWEBOK

El público al que va dirigida la Guía SWEBOK incluye:

- Organizaciones públicas y privadas: Aquellas que buscan establecer y promover una comprensión consistente de la ingeniería de software dentro de sus operaciones, particularmente al establecer estándares de educación y capacitación, clasificaciones laborales y políticas de evaluación del desempeño.
- Ingenieros de software en ejercicio: Personas que trabajan activamente en el campo de la ingeniería de software y que pueden utilizar la Guía como referencia para las mejores prácticas y el desarrollo profesional.

- Responsables de políticas (Auditores): Los involucrados en la creación de políticas públicas relacionadas con la ingeniería de software, incluido el desarrollo de reglas de licencia y pautas profesionales.
- Sociedades Profesionales: Organizaciones que definen estándares de acreditación para programas universitarios de ingeniería de software, así como reglas y pautas de certificación para profesionales en el campo.
- Estudiantes de ingeniería de software: Personas que estudian ingeniería de software que pueden utilizar la Guía para profundizar su comprensión de la disciplina.
- Educadores y capacitadores: Responsables de desarrollar planes de estudio y contenidos de cursos en programas de ingeniería de software.

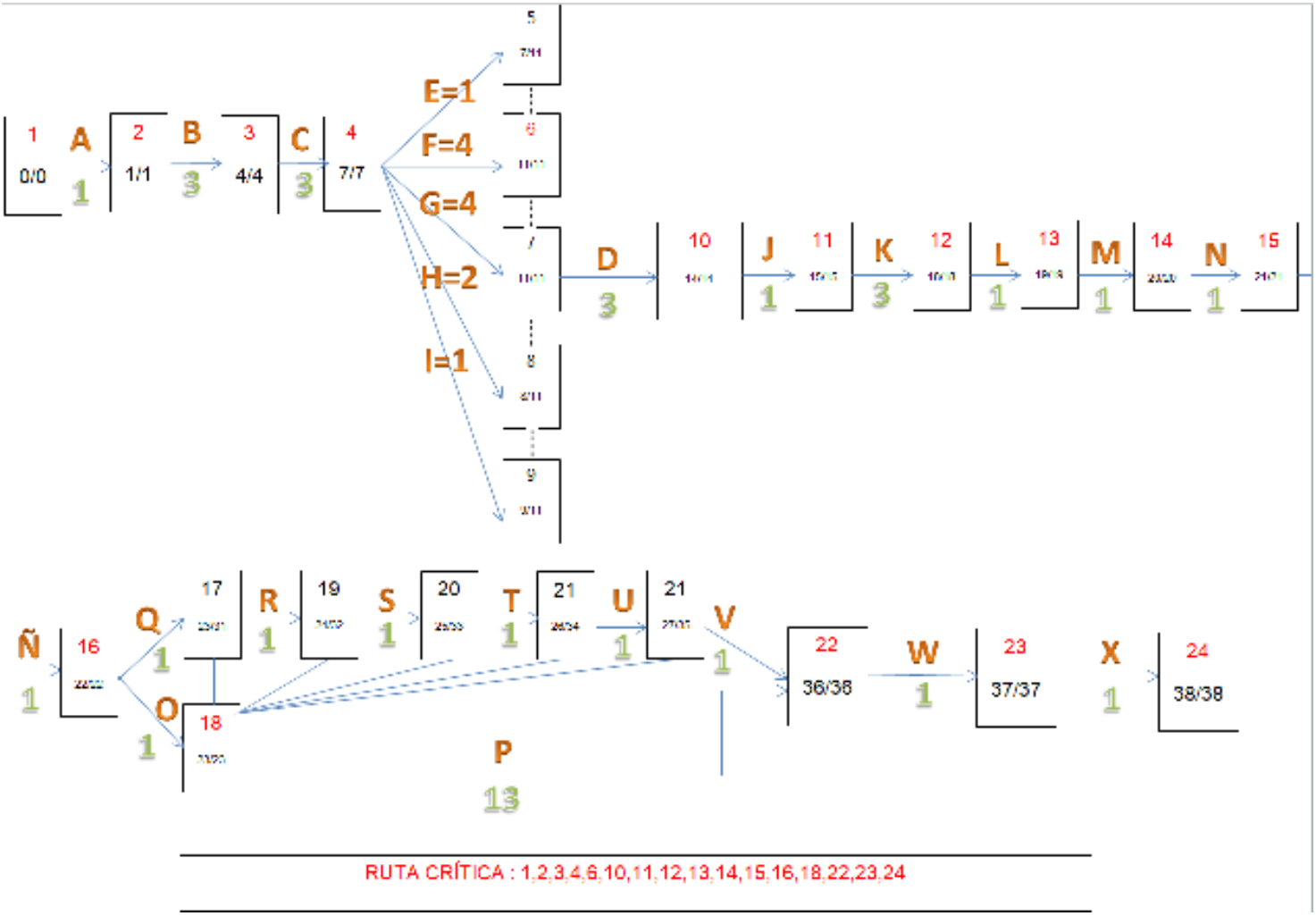
4. Planeación y Metodología

4.1 DIAGRAMA DE GANTT ANEXADO EN UN ARCHIVO EXCEL APARTE

4.2 Diagrama de PERT y Ruta Crítica

ACTIVIDAD	PRESEDENTE	TIEMPO (Días)
Inicio de Proyecto, presentación de rubrica (A)	-	1
Acuerdos con el equipo 2 para selección de tema (B)	A	3
Trabajo Preliminar (C)	B	3
Ensamblaje de elementos en html del Tutorial (D)	E,F,G,H,I	3
Primera cita con el profesor (E)	C	1
Producción del video (F)	C	4
Actualización de Contenidos (G)	C	4
Redacción del cuestionario (H)	C	2
Diseño de Sopa de Letras (I)	C	1
Revisión en Laboratorio (J)	D	1
Correcciones tras revisión (K)	J	3
Primera entrega de proyecto (L)	K	1
Recalendarización De Entrega (M)	L	1
Segunda Cita con el profesor (N)	M	1
Tercera Cita con el profesor (Ñ)	N	1
Plan de Corrección con Metodología KANBAN (O)	Ñ	1
Revisión de Contenidos para corrección (P)	O	12

Primer Informe de Avance (Q)	P	1
Segundo Informe de Avance (R)	Q	1
Tercer Informe de Avance (S)	R	1
Cuarto Informe de Avance (T)	S	1
Quinto Informe de Avance (U)	T	1
Sexto Informe de Avance (V)	V	1
Revisión Final (W)	P,V	1
Entrega Final (X)	W	1



4.3 Fases del Desarrollo (Metodología)

Para el desarrollo de nuestro tutorial utilizamos una metodología de Kanban donde anotábamos nuestras actividades en un documento en línea que podía ser consultado por cualquiera de nosotros y las actualizábamos usando 3 columnas para indicar su estado de desarrollo en ese momento como se presenta a continuación.

EJEMPLO DE KANBAN EN UN PUNTO EN CONCRETO DE NUESTRO DESARROLLO

TO DO	DOING	DONE
		PONER PRIMERO NOMBRE DE LA UNIVERSIDAD ORDEN ES : UNIVERSIDAD…. FACULTAD…. ASIGNATURA…
		En “TUTORIAL 2025-2 4” cambiar por “TUTORIAL 2025-2-1”
		REALIZAR VIDEO
		FECHAS AJUSTAR
	EN LA INTRODUCCIÓN , LO DEL DESARROLLO COLABORATIVO, MANTENIMIENTO CONTINUO E INGENIERÍA ESPECIALIZADA DESARROLLARLO, NO PONERLO EN EL DESCARGABLE TIENE QUE ESTAR EN LA PÁGINA	
	EN ÁREAS DE CONOCIMIENTO, AGREGAR LAS DESCRIPCIONES COMPLETAS DE CADA ÁREA	
	CUIDAR MAYÚSCULAS Y MINÚSCULAS	
	CUESTIONARIO DINÁMICO (SE DEBE PODER RESPONDER)	
	CONSISTENCIA EN USO DE MINUSCULAS Y MAYUSCULAS No se vale : ingeniería de Software	

	Es: Ingeniería de Software	
AGREGAR IMÁGENES		

5. Guión del Video Tutorial

Hola, ¿cómo están?
Somos el equipo 1 del grupo 04 de Ingeniería de Software de la Facultad de Ingeniería. Y en este vídeo te vamos a contar de forma sencilla qué es el famoso SWEBOK.
Sí, ya sé… el nombre suena medio raro. Pero créenos: si te interesa el desarrollo de software o te estás formando como ingeniero o ingeniera de software, esto te será muy útil.

¿Qué es el SWEBOK?

SWEBOK significa Software Engineering Body of Knowledge. Es una guía reconocida a nivel internacional que reúne el conocimiento aceptado y consensuado en la teoría y práctica de la Ingeniería de Software.
¿Y quién la respalda? Nada más y nada menos que la IEEE Computer Society, una de las organizaciones más grandes e importantes del mundo en tecnología e ingeniería.
Es un estándar oficial. No es un libro aburrido: es una herramienta poderosa que te organiza lo que ya sabes, lo que te falta y cómo puedes seguir creciendo.

¿Por qué es importante?

¿Quieres saber qué te falta por aprender? ¿Qué tan bien preparado estás? ¿O cómo se ve el camino completo de un proyecto de software?
El SWEBOK te ayuda con eso.
Sirve como una brújula: te muestra el panorama general, te ayuda a planear tu aprendizaje, y a hablar el mismo idioma que se usa en la industria.

¿Qué contiene el SWEBOK?

Desde su primera versión en 2004, el SWEBOK ha evolucionado.
La versión más reciente es la V4.0, publicada en 2024, y contiene 15 Áreas de Conocimiento, también llamadas Knowledge Areas (KAs).
Incluyen temas desde los requisitos y el diseño, hasta pruebas, mantenimiento, gestión, calidad, ética y fundamentos computacionales.
Además, en esta versión se refuerza el enfoque en ingeniería moderna, como el desarrollo ágil, la evolución de prácticas en la nube y la automatización.
Pero no te preocupes, no necesitas aprenderlas todas ya mismo.
En este video te vamos a contar las 5 más esenciales, las que forman el núcleo del desarrollo de software.

Las 5 áreas clave del SWEBOK V4.0

1. Requisitos del software

Aquí empieza todo. Antes de programar, hay que saber qué se necesita: ¿Quién usará el sistema? ¿Qué debe hacer? ¿Qué condiciones debe cumplir?

Se definen los requisitos funcionales (lo que el sistema debe hacer) y no funcionales (cómo debe hacerlo).

Una mala captura de requisitos puede hacer que todo lo demás falle.

2. Diseño del software

Una vez definidos los requisitos, se planea la estructura del sistema: la arquitectura, los módulos, sus interacciones, la interfaz.

Es como hacer planos antes de construir una casa.

Un buen diseño evita problemas a futuro y mejora la calidad del software.

3. Construcción del software

Aquí se programa: se escribe el código, se compila, se prueba.

En esta etapa se aplican principios de calidad, control de versiones y automatización.

También implica revisar si los requisitos y el diseño realmente funcionan al implementarlos.

4. Pruebas de software

¿Funciona bien? ¿Qué pasa si algo se rompe? ¿Se comporta como se espera?

Aquí hacemos pruebas unitarias, de integración, de sistema y de aceptación.

Una buena estrategia de pruebas reduce errores costosos más adelante.

5. Mantenimiento del software

El trabajo no termina al entregar. Hay que actualizar, corregir y adaptar el software a lo largo del tiempo.

Muchos proyectos dedican más tiempo al mantenimiento que al desarrollo inicial.

La V4.0 enfatiza también el mantenimiento en contextos de integración continua y despliegue automatizado.

¿Y por qué deberías conocer esto?

Porque estas cinco áreas son la base de cualquier proyecto serio de software.

Con el SWEBOK puedes identificar en qué parte del proceso estás, qué te falta por aprender, y cómo mejorar.

Por ejemplo:

¿Ya sabes programar? ¡Perfecto!

Pero… ¿sabes cómo captar bien los requisitos? ¿Has hecho pruebas automatizadas? ¿Has mantenido software real después de 6 meses?

El SWEBOK te da una visión completa. No se trata solo de codificar, sino de saber cómo construir software de calidad desde cero hasta su evolución continua.

¿Dónde puedes saber más?

Consulta la guía oficial en el sitio de la IEEE:

https://www.computer.org/education/bodies-of-knowledge/software-engineering

Ahí puedes ver ejemplos, leer más sobre las otras áreas, revisar mapas de competencias y consultar terminología profesional.

Conclusión

El SWEBOK V4.0 no es solo teoría: es una guía estructurada, práctica y reconocida que te acompaña a lo largo de tu formación y tu carrera profesional.

Ya sea que trabajes en equipos ágiles, en grandes organizaciones o estés empezando tu propio proyecto, entender el SWEBOK te hace un mejor ingeniero o ingeniera de software.

Espero que este video te haya servido para entenderlo mejor.

Mucha suerte en tus próximos proyectos.

¡Nos vemos pronto!

6. Actividades Didácticas

6.1 Sopa de letras

Áreas del SWEBOK

O	N	W	I	A	E	X	A	R	B	Ó	M	Ñ	R	Q	A	H	S	A	P
P	Ó	T	T	Í	H	M	D	G	I	C	Ñ	É	E	A	C	Z	O	Í	R
E	I	É	Ñ	R	D	S	M	H	W	I	K	R	Q	I	I	É	C	R	Á
R	C	M	V	E	G	Ó	N	A	B	T	B	H	U	P	T	O	I	E	C
A	A	A	S	I	J	M	A	Í	N	J	R	G	I	É	Á	D	T	I	T
C	R	N	P	N	I	O	Í	R	M	H	X	Z	S	Í	M	T	Á	N	I
I	U	T	N	E	Ñ	D	M	E	K	S	T	L	I	Z	R	A	M	E	C
O	G	E	Ó	G	C	E	O	I	U	L	Y	Ú	T	É	O	J	E	G	A
N	I	N	I	N	C	L	N	N	K	I	C	Ú	O	X	F	Ó	T	N	P
E	F	I	C	I	D	O	O	E	C	L	M	I	S	T	N	Z	A	I	R
S	N	M	C	E	A	S	C	G	U	A	G	G	Ó	N	I	A	M	E	O
Q	O	I	U	D	D	Y	E	N	U	W	L	O	W	É	I	P	É	D	F
T	C	E	R	N	I	M	Q	I	Á	G	J	I	Ñ	Í	C	H	Y	O	E
R	N	N	T	Ó	R	É	L	Á	N	Q	G	U	D	E	Z	D	O	S	S
M	Ó	T	S	I	U	T	H	C	S	S	F	U	R	A	S	Ü	K	E	I
É	I	O	N	T	G	O	K	S	A	B	E	U	R	P	D	I	É	C	O
F	T	Q	O	S	E	D	Ú	T	O	Q	S	Z	Y	Í	Z	V	D	O	N
W	S	P	C	E	S	O	I	Ú	D	C	Á	Ñ	Ú	É	R	E	Z	R	A
Ü	E	P	P	G	S	S	Á	B	Ü	Í	G	E	T	L	C	F	C	P	L
R	G	Í	A	R	U	T	C	E	T	I	U	Q	R	A	M	Í	W	R	D

- REQUISITOS
- ARQUITECTURA
- DISEÑO
- CONSTRUCCIÓN
- PRUEBAS
- OPERACIONES
- MANTENIMIENTO
- GESTIÓNCONFIGURACIÓN
- GESTIÓNDEINGENIERÍA
- PROCESODEINGENIERÍA
- MODELOSYMÉTODOS
- CALIDAD
- SEGURIDAD
- PRÁCTICAPROFESIONAL
- ECONOMÍA,
- INFORMÁTICA
- MATEMÁTICOS
- INGENIERÍA

6.2 Cuestionario

1.- ¿Que significan las siglas SWEBOK?

A) System Elemental Book of Knowledge B) Software Engineering Building Knowledge

C) Software Engineering Body of Knowledge D) System of Engineering for Basic Keywords

C) Software Engineering Body of Knowledge

2.- ¿Cuáles de las siguientes áreas NO se contemplan?

A) Requerimientos del software C) Mantenimiento del Software

B) Diseño del Software D) Ensamblado del Software

D) Ensamblado del Software

3.- Cuales de los siguientes pares representan 2 públicos objetivos para la guía SWEBOK

A) Organizaciones públicas, Aseguradoras

B) Ingenieros de software en ejercicio, Responsables de políticas (Auditores)

C) Sociedades Profesionales, Reguladores

D) Estudiantes de ingeniería de software, Creadores de Contenido

B) Ingenieros de software en ejercicio, Responsables de políticas (Auditores)

4.- Cuales de los siguientes NO es uso de la guía SWEBOK desde la perspectiva de un ingeniero de software

A) Como una métrica para evaluar el software producido así como para regular los estándares que se apliquen al mismo.

B) Comprobar sus propios conocimientos y experiencia en cada AC (Área de Conocimiento).

C) Obtener más información de las referencias y enlaces de cada AC, así como de los temas relacionados y emergentes.

D) Como una guía para comprender y utilizar términos, conceptos, mejores prácticas y estándares de ingeniería de software.

A) Como una métrica para evaluar el software producido así como para regular los estándares que se apliquen al mismo.

5.- ¿Qué nivel de certificación SWEBOK es la máxima alcanzable por una empresa desarrolladora de software?

A) Nivel Uno B) Nivel Dos C) Nivel Tres D) No existe

D) No existe

6.- ¿Cuál es una opción de certificación válida en lugar del CSDP?

A) CMMI B) ISO 15504) C) IEEE CSDA y CSE D) IPPD

C) IEEE CSDA y CSE

7.- ¿Cuál es el concepto que se refiere a la creación de software mediante una combinación de codificación, verificación, pruebas unitarias, pruebas de integración, y depuración?

A) Construcción Del Software B) Depuración Del Software C) Validación Del Software D) Refinación del Software

A) Construcción de software

8.- Proceso que se compone de la verificación dinámica del comportamiento de un programa con un conjunto finito de casos de pruebas

Análisis De Software B) Pruebas de software C) Casos de Uso D) Persistencia del Software

B) Pruebas de software

9.- ¿Cuál de los siguientes fue un problema principal que llevó a la "Crisis del Software" en la década de 1960?

A) Políticas Gubernamentales B) Capitales Insuficientes C) Equipo Deficiente D) Falta de Metodologías Estructuradas

D) Falta de Metodologías Estructuradas

10.- ¿Cómo influyó la revolución de Internet en la evolución de las metodologías de desarrollo de software en la década de 1990?

A) Proporciono métodos de trabajo más simples que priorizaban la satisfacción del cliente y mejoraban la comunicación entre los distintos departamentos de un equipo de desarrollo.

B) Permitió que las metodologías tradicionales fueran optimizadas.

C) Dio origen al surgimiento de metodologías ágiles como Scrum y Extreme Programming (XP), que priorizan la colaboración, la adaptabilidad y la entrega continua de software funcional.

D) Las metodologías crecieron en complejidad pero también aumento su capacidad de atender a desafíos de software más extensos con un resultado más predecible y confiable con respecto al pasado.

C) Dio origen al surgimiento de metodologías ágiles como Scrum y Extreme Programming (XP), que priorizan la colaboración, la adaptabilidad y la entrega continua de software funcional.

7. Referencias

7.1 Bibliografía

1. 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.

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

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

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

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

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

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

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

7.2 Documentos electrónicos

1. IEEE Computer Society. (2014). Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), Version 4.0. IEEE. https://www.computer.org/education/bodies-of-knowledge/software-engineering

2. easymaint.net. (s.f.). Explorando el Mantenimiento de Software: Garantizando la Salud y Eficiencia de tus Aplicaciones Digitales. Obtenido de https://www.easymaint.net/faq/que-es-el-mantenimiento-de-software.html#:~:text=Definici%C3%B3n%20de%20Mantenimiento%20de%20Software:%20El%20mantenimiento,correctivo)%2C%20la%20implementaci%C3%B3n%20de%20nuevas%20caracter%C3%ADsticas%20

3. Hironori Washizaki, M.-I. S.-S. (s.f.). An Overview of the SWEBOK Guide. Obtenido de https://sebokwiki.org/wiki/An_Overview_of_the_SWEBOK_Guide#Introduction

4. Society, I. C. (s.f.). Software Engineering Body of Knowledge (SWEBOK). Obtenido de https://www.computer.org/education/bodies-of-knowledge/software-engineering

5. UNAM, D. G. (s.f.). Software Engineering Body of Knowledge (SWEBOK). Obtenido de https://www.red-tic.unam.mx/content/software-engineering-body-knowledge-swebok

6. Univeritat Oberta de Catalunya por Viladrosa, R. C. (s.f.). ¿Qué debes saber para desarrollar software de forma profesional? Obtenido de https://blogs.uoc.edu/informatica/es/que-debes-saber-para-desarrollar-software-de-forma-profesional/

9.