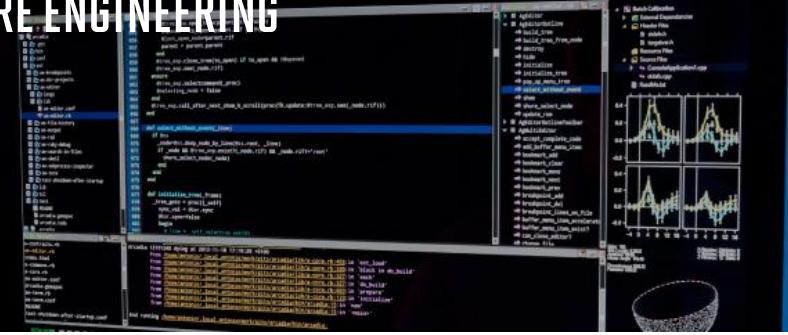


# SOFTWARE ENGINEERING



## ☰ Objetivo

## ☰ Introducción

## ☰ Software Product

### ?

## Actividad 1

## ☰ Culture & Lifecycle

### ?

## Actividad 2

## ☰ Requirements

### ?

## Actividad 3

## ☰ Resumen

## ☰ Referencias

## ☰ PDF descargable

## OBJETIVO

---

Al finalizar la unidad de aprendizaje, el estudiante ejecuta el Lean UX Process, identificando la problemática para un dominio y contexto determinado, estableciendo supuestos y las declaraciones de hipótesis que definen la solución.



## INTRODUCCIÓN

---

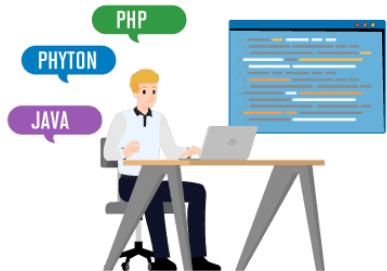
### SOFTWARE

- Es más que código de programación. Un programa (*program*) es un código ejecutable que sirve para algún propósito computacional.
- Se considera software a una colección de programas ejecutables, bibliotecas (*libraries*) asociadas y documentación.
- El software, cuando se elabora para requisitos específicos, se denomina producto de software (*software product*).

### ENGINEERING (INGENIERÍA)

- Abarca lo relacionado con el desarrollo de productos, utilizando principios y métodos científicos claramente definidos.

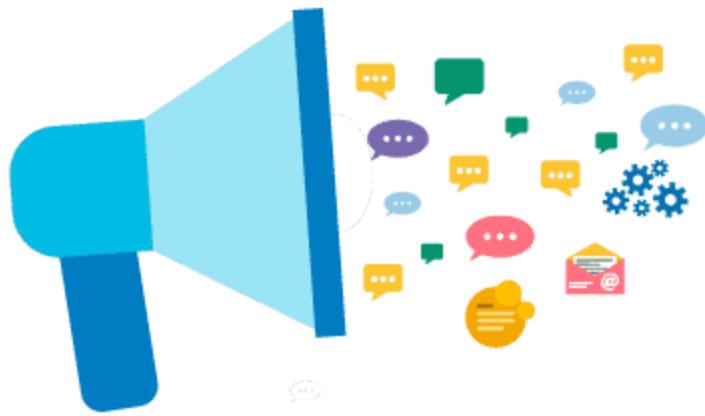
### SOFTWARE ENGINEERING (INGENIERÍA DE SOFTWARE)



Es la rama de ingeniería asociada con el desarrollo de productos de software utilizando principios, métodos y procedimientos científicos claramente definidos.



El outcome de ingeniería de software es un producto de software eficiente y confiable.



IEEE brinda definiciones claras sobre Software Engineering, cubriendo dos perspectivas: la de la aplicación o puesta en práctica de aproximaciones para la ingeniería de software, así como el estudio de dichos approaches.

1

The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software.

2

The study of approaches as in the above statement.

Fritz Bauer, un pionero alemán de Computer Science, nos trae en 1972 una definición de Software Engineering.



Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and work efficiently on real machines.

**Fritz Bauer, a German computer scientist**

**i** LO INTERESANTE DE ESTA DEFINICIÓN, ES QUE NO SOLO ABORDA LA FUNCIONALIDAD Y EFICIENCIA DEL SOFTWARE EN ESCENARIOS REALES, SINO LA CONSIDERACIÓN DE LA ECONOMÍA EN SU DESARROLLO.

## SOFTWARE PRODUCT

---



En relación a quién va dirigido (Target) el software, se puede identificar dos categorías:



## **GENERIC PRODUCTS (PRODUCTOS GENÉRICOS)**

### **DEFINICIÓN**

Los productos de software genéricos son sistemas stand-alone (independientes), producidos por organizaciones de desarrollo y vendidos en el mercado abierto a cualquier cliente capaz de adquirirlos.

### **EJEMPLOS**

Como ejemplos de productos genéricos, se pueden considerar los siguientes: gestión de bases de datos, entornos de procesamiento de software para computadoras personales o de texto, herramientas para arte, dibujo y animación; así como herramientas para gestión de proyectos.

## **CUSTOMIZED PRODUCTS (PRODUCTOS PERSONALIZADOS)**

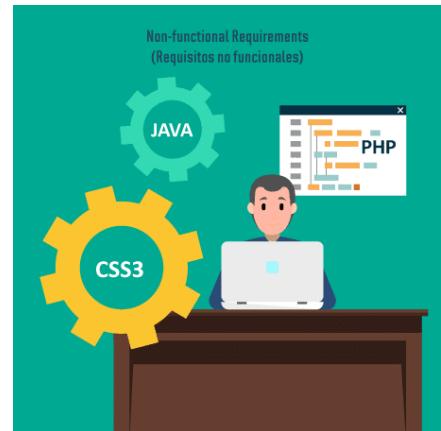
### **DEFINICIÓN**

Los productos de software personalizados son sistemas encargados por algún cliente. Un software developer / contractor desarrolla el software especialmente para dicho cliente acorde con los requirements (requisitos) del cliente.

### **EJEMPLOS**

Por ejemplo, se puede incluir en esta categoría a los sistemas de control para dispositivos electrónicos, software para banca, sistemas de software para comunicación por voz, sistemas desarrollados para brindar soporte a procesos de negocio particulares y sistemas de control de tráfico aéreo.

En relación al propósito que cumple el software, se pueden identificar las siguientes categorías:



## APPLICATION SOFTWARE (SOFTWARE DE APLICACIÓN)

Software creado para un propósito específico. Generalmente, es una colección de programas utilizados por usuarios finales. Puede referirse a Application Software como Application (Aplicación) o simplemente app. Este software ayuda a los usuarios a realizar tareas específicas.

Algunos ejemplos incluyen:

- Word processing software (Software de procesamiento de textos)
- Database programs (Programas de bases de datos)

- Entertainment software (Software de entretenimiento)
- Business software (Software para negocios)
- Educational software (Software educativo)
- Computer-aided design (CAD) software (Diseño asistido por computadora)
- Spreadsheet software (Software para hojas de cálculo)

ⓘ FUENTE: [HTTPS://WWW.DEFIT.ORG/APPLICATION-SOFTWARE/](https://www.defit.org/application-software/)

2

## LIBRARY (BIBLIOTECA)

1

2

3

Proporciona un conjunto de funciones/objetos/módulos de apoyo (helper), los cuales son llamados por la aplicación para alguna funcionalidad específica.

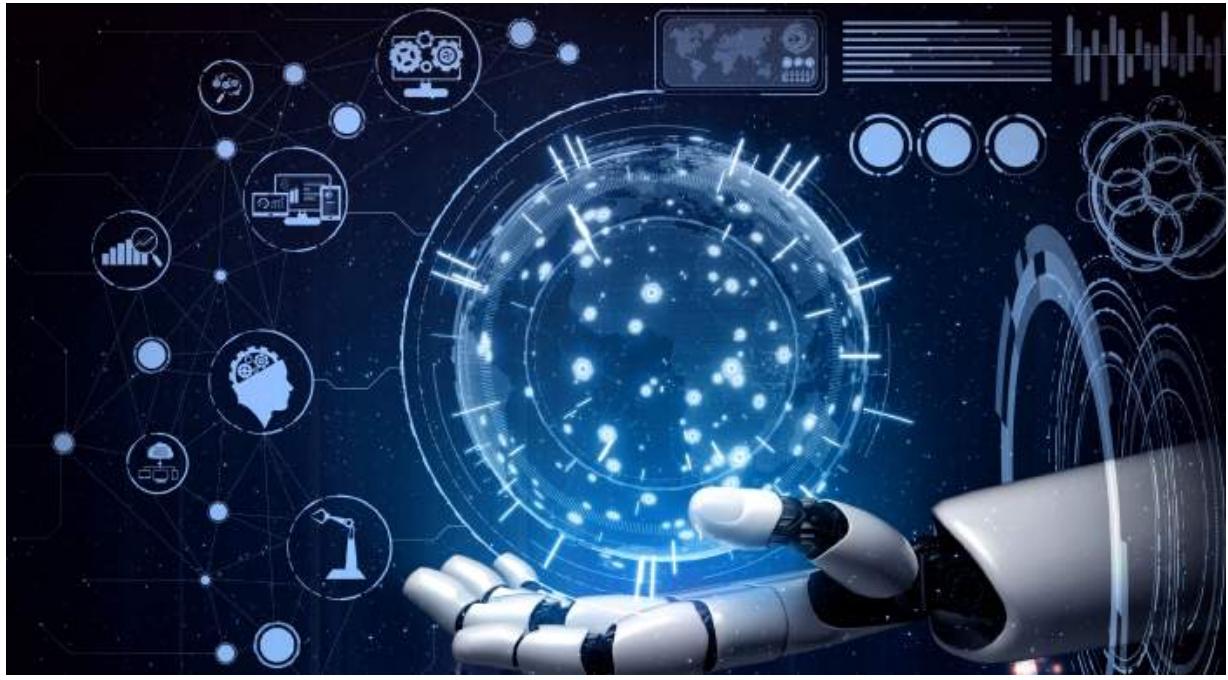


1

2

3

Normalmente, se enfocan en un aspecto específico (por ejemplo, strings, I / O, sockets), de tal forma que sus API (Application Programming Interface) deben ser más pequeños y requieren menos dependencias.



1

2

3

En el caso del paradigma orientado a objetos, serían una colección de definiciones de clases.



## ¿Por qué son necesarias?

Por una razón muy simple, pero también muy importante, code reuse (reutilización de código).

3

FRAMEWORK

---

**“A software framework is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software”.**

#### DEFINICIÓN 1

Esta abstracción incluye funciones abiertas o no definidas u objetos que el usuario escribe para crear una aplicación personalizada.

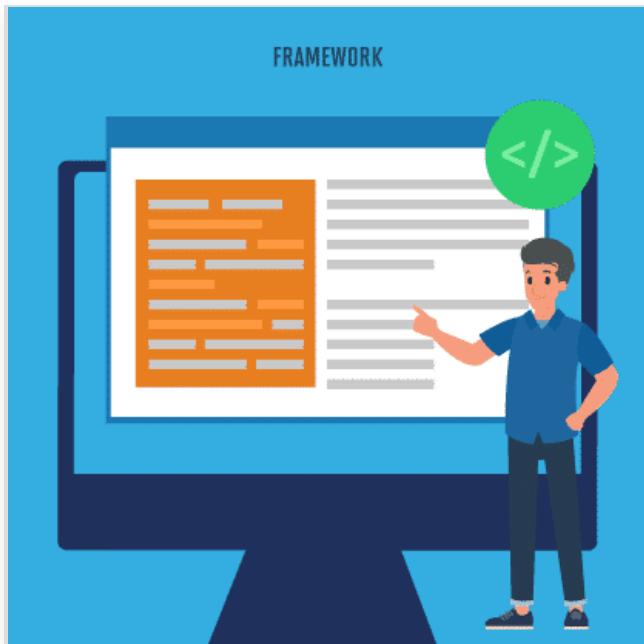
#### DEFINICIÓN 2

Dado que un framework es en sí una aplicación, tiene un alcance mayor e incluye casi todo lo necesario para crear una aplicación según las necesidades del usuario.



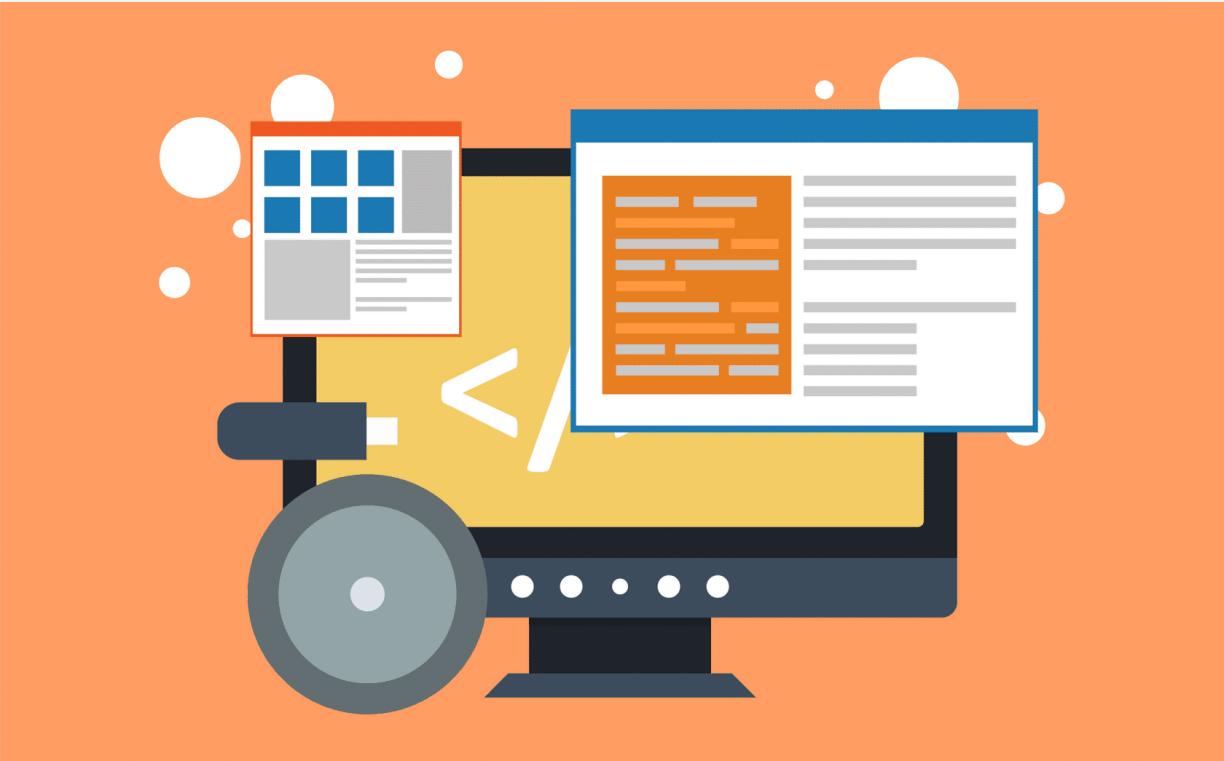
Realiza un conjunto de operaciones específicas y bien definidas. Ejemplos: Protocolos de red, compresión, manipulación de imágenes, utilidades para cadenas, evaluación de expresiones regulares, matemáticas, etc.

1 of 2



Se define como un esqueleto donde la aplicación define el contenido de la operación completando el esqueleto. Ejemplos de frameworks: Web application system, Plug-in manager, GUI system. El framework define solo el

2 of 2



## INVERSION OF CONTROL

Cuando invocamos un método en una biblioteca, estamos en control. En un framework, por el contrario, el control se invierte; es decir, el framework invoca a nuestro código.



FUENTE: [HTTPS://WWW.GEEKSFORGEEKS.ORG/SOFTWARE-FRAMEWORK-VS-LIBRARY/](https://www.geeksforgeeks.org/software-framework-vs-library/)

## ACTIVIDAD 1

---

Lee los siguientes enunciados y marca la respuesta correcta:

---

*Pregunta*

**01/04**

Los productos de software genéricos son sistemas independientes producidos por organizaciones de desarrollo y vendidos en el mercado abierto a cualquier cliente capaz de adquirirlo. ¿A qué término pertenece esta definición?

---

- Generic products
- Customized products
- Library
- Framework

*Pregunta*

**02/04**

Los productos de software personalizados son sistemas encargados por algún cliente. Un software contractor desarrolla el software especialmente para dicho cliente, acorde con los requisitos del cliente. ¿A qué término pertenece esta definición?

---

- Generic products
- Customized products
- Library
- Framework

*Pregunta*

**03/04**

El \_\_\_\_\_ proporciona un conjunto de funciones/objetos/módulos de apoyo, llamados por la aplicación para alguna funcionalidad específica.

---

Library

Framework

Software

Customized

*Pregunta*

**04/04**

Son definiciones del Framework:

---

- Incluye funciones abiertas o no definidas u objetos que el usuario escribe para crear una aplicación personalizada.
- Es una aplicación que tiene un alcance mayor e incluye casi todo lo necesario para crear una aplicación según las necesidades del usuario.
- Realiza un conjunto de operaciones específicas y bien definidas. Ejemplos: Protocolos de red, compresión, manipulación de imágenes, utilidades para cadenas, evaluación de expresiones regulares, matemáticas, etc.
- Son productos de software genéricos con sistemas independientes que son producidos por organizaciones de desarrollo y contenido digital.

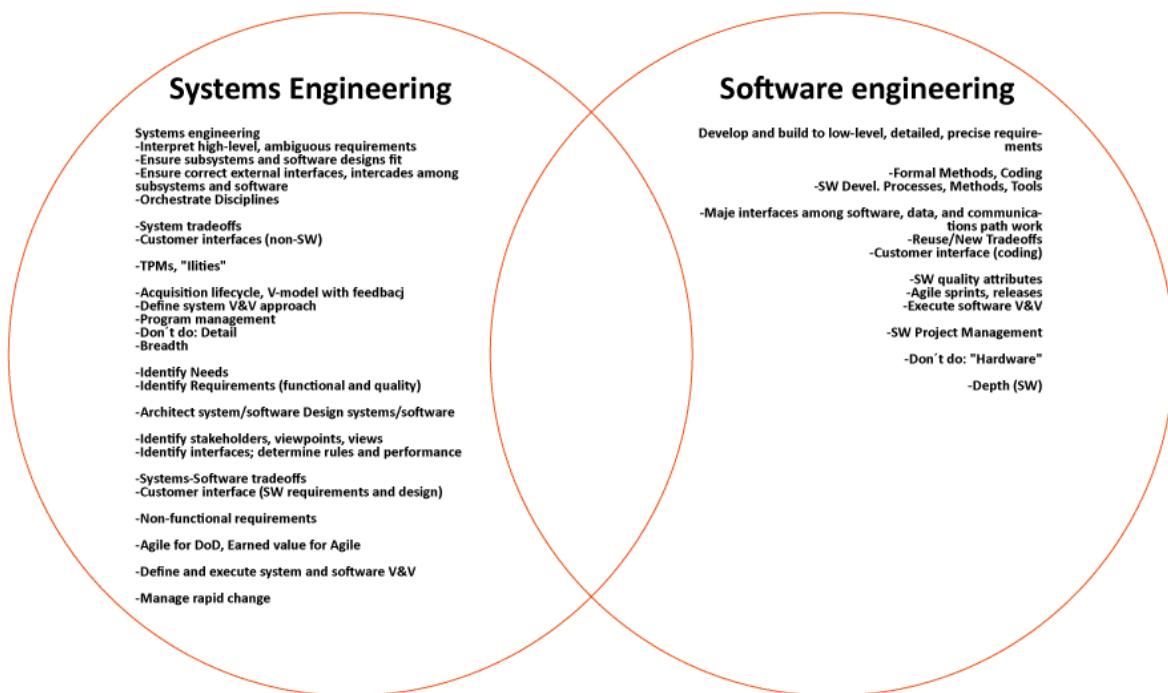
## CULTURE & LIFECYCLE

---



En el siguiente gráfico se aprecian las actividades coincidentes entre Systems engineering (Ingeniería de Sistemas) y Software Engineering (Ingeniería de Software).

# Software Engineering and Systems Engineering



- Se puede notar que las principales coincidencias están relacionadas con Requirements (Requisitos).
- Esto ayuda a entender la importancia del área de conocimiento, y cómo el dominio de conceptos y prácticas favorece la comunicación y colaboración entre otras ramas de ingeniería.

**SWEBOK**

La guía para el Software



Engineering Body of Knowledge (SWEBOK Guide) describe el conocimiento generalmente aceptado sobre ingeniería de software. Sus 15 Knowledge Areas (áreas de conocimiento) o KAs resumen los conceptos

El siguiente gráfico muestra la organización conceptual del *Guide to the Software Engineering Body of Knowledge* (SWEBOK Guide). Entre los tópicos principales, se encuentra el Software Requirements (Requisitos de Software).

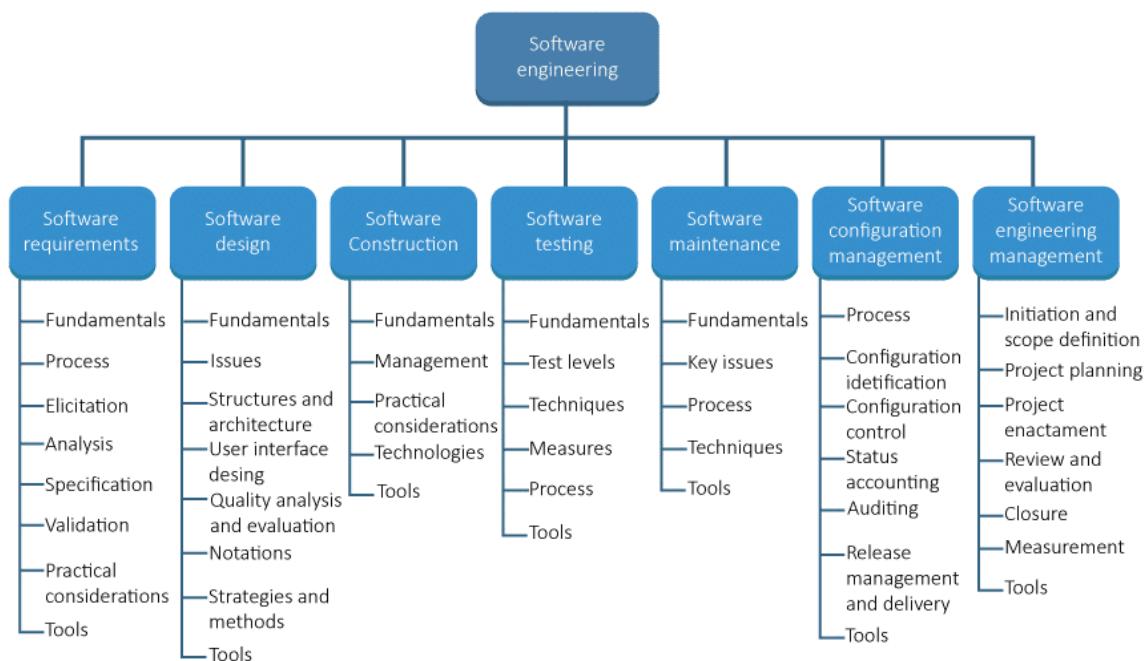


FIGURE 1. Part of the conceptual layout of the Guide to the Software Engineering Body of Knowledge (SWEBOK Guide). The boxes show major topics, with subtopics listed in the descending structures.

 FUENTE: [HTTPS://IEEEXPLORE.IEEE.ORG/STAMP/STAMP.JSP?TP=&ARNUMBER=7367983](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7367983)

---

Enmarcados en este tópico aparecen aspectos importantes que se irán tratando a lo largo del curso, como por ejemplo, las fases de Elicitation (Obtención), Analysis (Análisis), Specification (Especificación) y Validation (Validación).

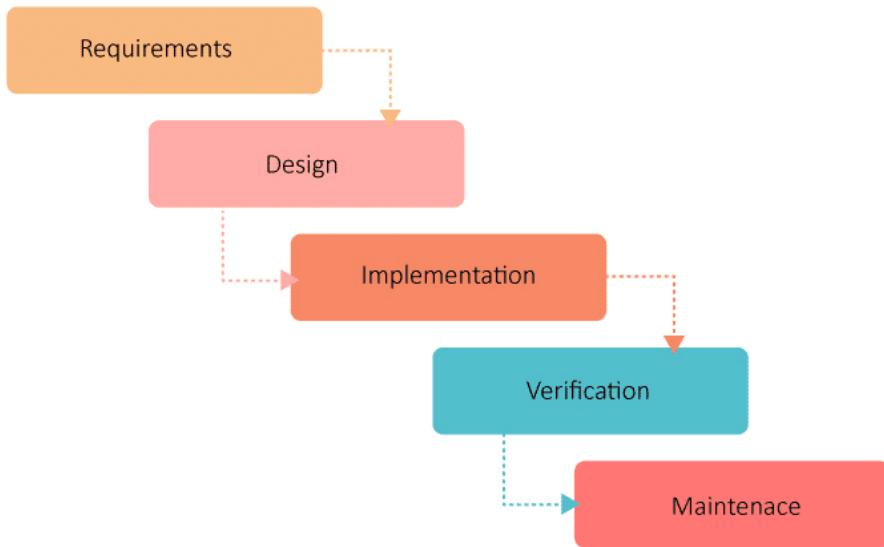
---



1

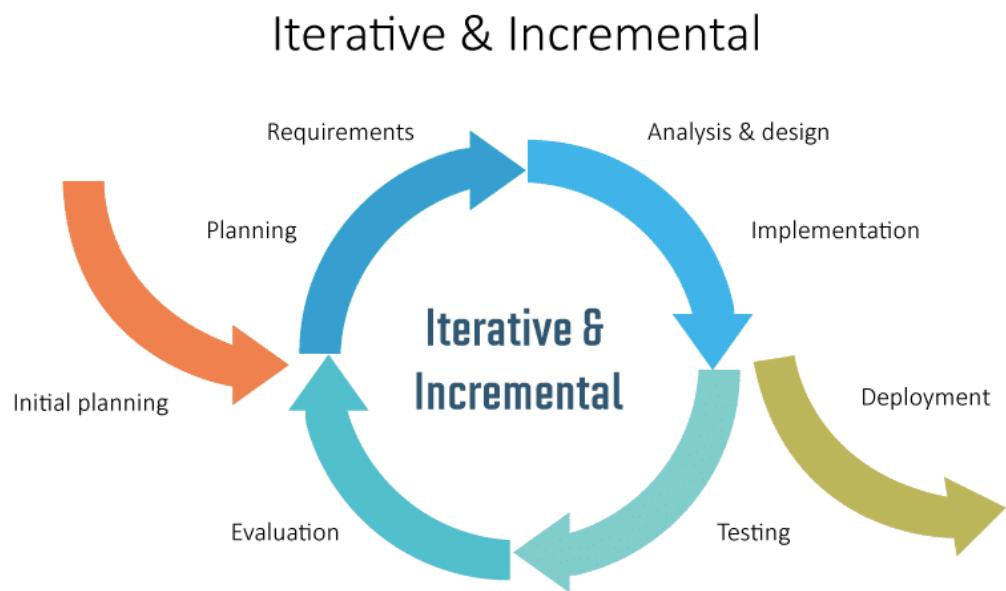
## **SOFTWARE DEVELOPMENT LIFECYCLE (CICLO DE VIDA DE DESARROLLO DE SOFTWARE)**

## Waterfall



Waterfall Model (Modelo en Cascada) es uno de los enfoques clásicos en desarrollo de software, el cual describe un método de desarrollo lineal y secuencial. Bajo este enfoque, las fases están definidas por tareas y objetivos.

La característica fundamental de este enfoque es que una vez finalizada una fase (sí, debe finalizar primero), los resultados de ésta sirven de entrada para la siguiente.



1

En el Iterative & Incremental SDLC, la alta incertidumbre para la planificación se aborda realizando iteraciones, que son básicamente una división del proyecto en fases cíclicas en las que el proyecto va avanzando de forma progresiva.

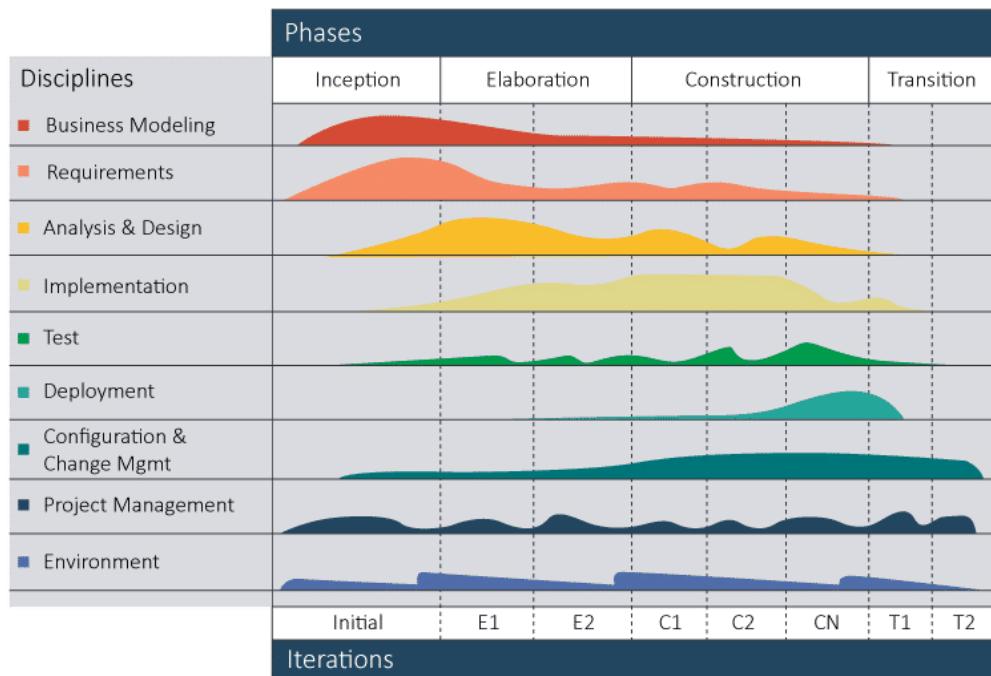
2

A cada ciclo se le denomina *iteración*. Cuando se combina los términos iterativo e incremental, cada ciclo de vida que se realiza va logrando concretar una porción completa del producto, servicio o resultado.

3

Cada porción generada por una iteración recibe el nombre de *incremento*. Por lo anterior se deduce que va produciendo porciones del resultado del proyecto acabadas en un 100%, iterando hasta tener todas las porciones llegando así al resultado esperado.

## Unified Process



Unified Process, inicialmente propuesto por Rational, empresa que fue adquirida por IBM, es un marco para el proceso de software, que sigue el ciclo de vida iterativo e incremental, pero agregando el concepto de fases:

- Inception (Inicio)
- Elaboration (Elaboración)
- Construction (Construcción)
- Transition (Transición)

Cada una de estas fases incluye una o más iteraciones (lo cual brinda la oportunidad de perfeccionar con el feedback del usuario). En cada fase puede refinarse los objetivos de las fases anteriores.

Dentro de cada fase y en cada iteración en particular, se lleva a cabo un ciclo de vida Waterfall (cascada), aunque dependiendo de la fase, los resultados no incluyen necesariamente software construido, llegando a otros artefactos o a prototipos.

## ACTIVIDAD 2

---

Lee los siguientes enunciados y marca la respuesta correcta:

---

*Pregunta*

**01/03**

¿Qué es SWEBOK?

---

- Ayuda a entender la importancia del área de conocimiento.
- Proporciona un conjunto de funciones, objetos y módulos de apoyo.
- Describe el conocimiento generalmente aceptado sobre ingeniería de software.
- Ninguna de las anteriores

*Pregunta*

**02/03**

El \_\_\_\_\_ es uno de los enfoques clásicos en desarrollo de software, el cual describe un método de desarrollo lineal y secuencial. Bajo este enfoque, las fases están definidas por tareas y objetivos.

---

- Framework
- Software Engineering
- Library
- Waterfall Model

*Pregunta*

**03/03**

Unified Process, es un marco para el proceso de software, que sigue el ciclo de vida iterativo e incremental, pero agregando el concepto de las siguientes fases:

---

Inception

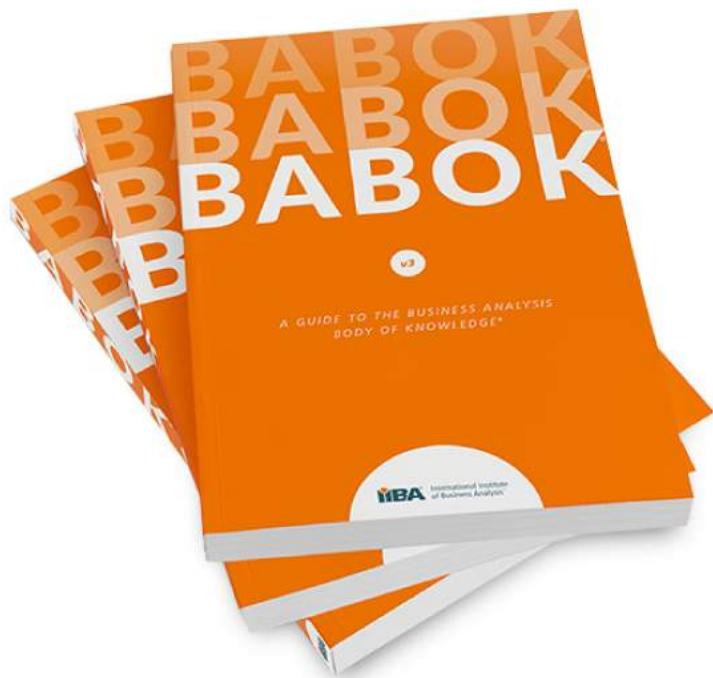
Elaboration

Construction

Testing

## REQUIREMENTS

---



El Business Analysis Body of Knowledge (Cuerpo de conocimiento de Análisis de Negocio) Guide, contiene una descripción de prácticas generalmente aceptadas en relación al Análisis de Negocio.

---



Se describen las áreas de conocimiento del Análisis de Negocios, sus actividades y tareas asociadas.

---



Por otro lado, también indica las habilidades necesarias para una ejecución efectiva.

---

#### IMPORTANCIA

En BABOK se define el Requirement (Requisito), lo cual permite entender la importancia de conocer los conceptos y saber aplicar las prácticas relacionadas con el mismo.

#### NEXO DIRECTO

El concepto de Requisito trasciende la Ingeniería de Software o la Ingeniería de Sistemas, estableciendo un nexo directo con el Análisis de Negocio.

## ALCANCE

Al revisar la definición de BABOK sobre Requirement, se puede entender en mejor manera su alcance y valor. Para comenzar, se considera una representación de necesidades.

## PROCESO

Como proceso, se enfoca en entender qué valor se puede entregar si el requisito se satisface. La naturaleza de la representación depende del contexto, pudiendo verse reflejada por escrito en documentos.

BABOK considera 4 tipos de Requisitos:



## BUSINESS REQUIREMENTS



Los Requisitos del Negocio enuncian metas, objetivos y outcomes con el fin de entender en mejor medida el porqué de un cambio y qué buscamos con el.

Estos pueden relacionar a la empresa como un todo, un área o hasta una iniciativa en particular. Los Business Requirements están principalmente relacionados con objetivos de alto nivel.

## EJEMPLO

Una empresa desea mejorar la eficiencia y el soporte en sus procesos relacionados con clientes y comercialización, llegando a necesitar un soporte de automatización de los procesos relacionados con CRM (Customer Relationship Management), Ventas y Marketing.

2

## STAKEHOLDER REQUIREMENTS



Están más relacionados con necesidades expresadas por los grupos de interés, las cuales van a permitir que se logre satisfacer los requisitos del negocio.

---

Al ser planteados por los grupos de interés, pueden ser resultado de la percepción de los mismos dentro del ámbito de las actividades que realizan en el negocio, los procesos en los que participan o supervisan, así como el ámbito de información que se utiliza en ese contexto.

**ⓘ SIN EMBARGO, ES IMPORTANTE RECONOCERLOS, PUES SON UN PUENTE ENTRE LOS BUSINESS REQUIREMENTS Y LOS SOLUTION REQUIREMENTS.**

#### EJEMPLO

---

El gerente de servicio al cliente puede solicitar un mecanismo para monitorear el tiempo de respuesta de cada solicitud de soporte de cliente de manera diaria para mejorar el tiempo de respuesta.

3

## SOLUTION REQUIREMENTS



Se orienta más a capacidades o atributos de calidad de la solución, a fin de satisfacer los Stakeholder Requirements.

---

Estos tipos de requisitos son los que permiten, por ejemplo, a los Ingenieros de Software (si la solución fuera crear un producto de software), contar con el nivel de detalle necesario para el desarrollo e implementación de una solución.

Los Solution Requirements pueden ser de dos tipos:



## FUNCTIONAL REQUIREMENTS

### DEFINICIÓN

Capacidades que la solución debe poseer en términos del comportamiento e información que ella debe manejar.

### EJEMPLO

- Registrar un usuario
- Realizar una compra en línea

# NON-FUNCTIONAL REQUIREMENTS

## DEFINICIÓN

Requisitos no relacionados directamente con el comportamiento de la solución, sino que describen condiciones bajo las cuales ésta debe mantenerse efectiva o cualidades que ésta debe tener, incluyendo aspectos como reliability, availability, portability, scalability, usability, maintainability.

## EJEMPLO

- Cada página debería cargar en máximo 2 segundos.

# TRANSITION REQUIREMENTS



Pueden estar relacionadas con la solución misma o con las condiciones que esta debe cumplir para que se pueda llevar a cabo la transición hacia el nuevo estado del negocio, el cual cuenta con la solución.



Estos requisitos, muy importantes de hecho, por lo general tienen una naturaleza temporal, pudiendo implicar, luego de la transición de estado, que se descarten productos o artefactos creados para satisfacerlos.

**(i) AYUDAN A PERMITIR LA IMPLEMENTACIÓN EXITOSA DE UN PROYECTO ENMARCADOS EN PERIODOS ESPECÍFICOS.**

EJEMPLOS

- Los usuarios deben estar entrenados para poder utilizar el sistema eficientemente.
- Los datos de años anteriores deben migrarse al sistema para generar reportes comparativos.

7

A continuación, algunos ejemplos de Requirements (requisitos) a nivel de Business, Stakeholder, Solution o Transition.

## Requirements

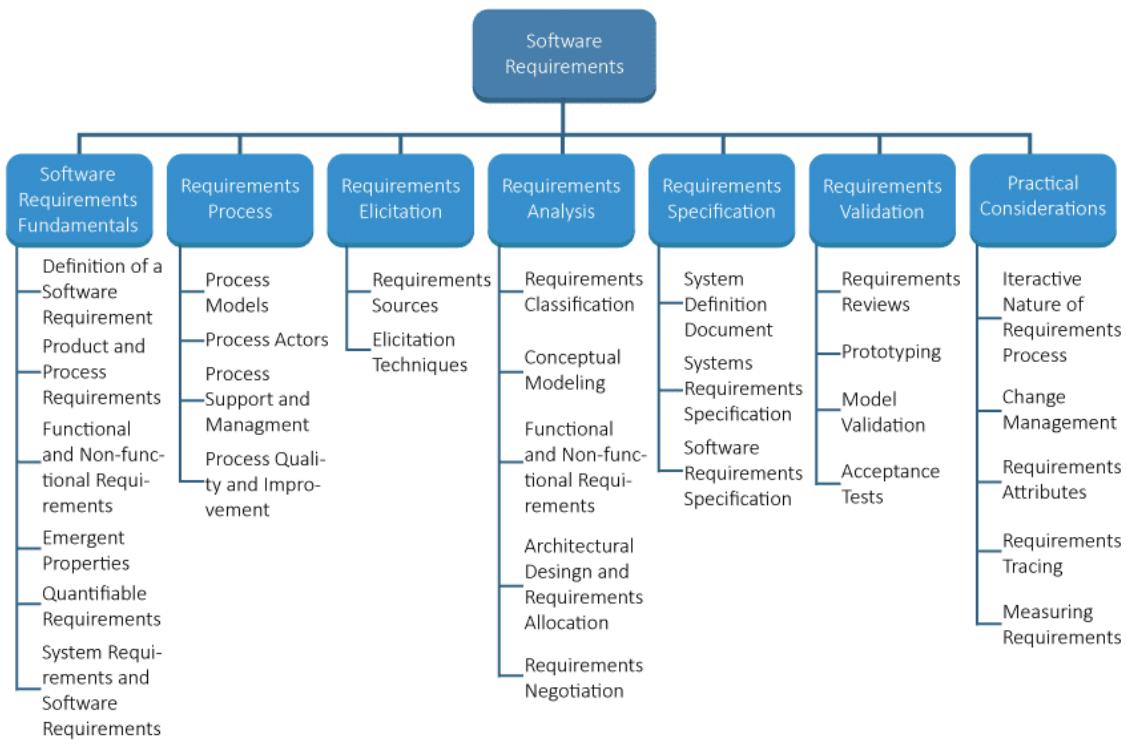
Type	Examples
<b>Business</b>	<ul style="list-style-type: none"><li>• Reducir las órdenes procesadas de forma incorrecta en 50% para el siguiente trimestre.</li><li>• Incrementar la repetición de órdenes de cliente en 10% en los seis meses posteriores al despliegue.</li></ul>
<b>User/Stakeholder</b>	<ul style="list-style-type: none"><li>• Agregar nueva cuenta de cliente.</li><li>• Ver el listado de órdenes.</li><li>• Verificar el estado de la orden.</li><li>• Crear una nueva orden.</li></ul>
<b>Functional/Solution</b>	<ul style="list-style-type: none"><li>• Mostrar apellido del cliente como enlace hacia la información de la cuenta.</li><li>• Permitir ordenar por fecha de apertura de cuenta.</li></ul>
<b>Non-functional</b>	<ul style="list-style-type: none"><li>• Permitir hasta 200 usuarios concurrentes.</li><li>• Requerir strong passwords de al menos 8 caracteres de longitud, contenido como mínimo un carácter no alfabético.</li></ul>
<b>Transition</b>	<ul style="list-style-type: none"><li>• Se debe contar con 50% de staff de soporte adicional durante el primer mes de operación del sistema, para atender la demanda de consultas que pudieran presentarse.</li></ul>

Al observar el gráfico, este resume los tópicos relacionados con el área de conocimiento de Requirements en el SWEBOK, se entiende que es un territorio amplio, a la vez importante para los procesos de Ingeniería.

Parte muy importante de la solución, reside en identificar y entender adecuadamente el problema.

En el gráfico se resaltan términos como Elicitation, Analysis, Specification y Validation, los cuales dan sentido y se abordarán más adelante en el curso.

## **SWEBOK**



## ACTIVIDAD 3

---

Lee los siguientes enunciados y marca la respuesta correcta:

---

*Pregunta*

**01/03**

¿Qué es BABOK?

---

- Ayuda a entender la importancia del área de conocimiento.
- Permite entender la importancia de conocer los conceptos y saber aplicar las prácticas relacionadas con el mismo.
- Describe el conocimiento generalmente aceptado sobre ingeniería de software.
- Ninguna de las anteriores

*Pregunta*

**02/03**

El BABOK se divide en 4 tipos de requisitos:

---

- Business requirements, solution requirements, transition requirements y stakeholder requirements.
- Transition requirements, stakeholder requirements, library requirements y software requirements.
- Solution requirements, transition requirements, stakeholder requirements y analysis requirements.
- Ninguna de las anteriores

*Pregunta*

**03/03**

El \_\_\_\_\_ son tipos de requisitos con los que permiten, por ejemplo, a los Ingenieros de Software contar con el nivel de detalle necesario para el desarrollo e implementación de una solución.

---

- Business requirements
- Solution Requirements
- Stakeholder requirements
- Transition requirements

## RESUMEN

---



Reflexionemos sobre lo que esto hemos aprendido:

1

¿Qué es la Ingeniería de Software?

2

¿Cuándo hablamos de un Software Product y qué tipos de software existen?

**3**

¿Qué es el SWEBOk y por qué es importante?

**4**

¿Qué es SDLC y que enfoques existen para llevarlo a cabo?

**5**

¿Qué son Requirements y que tipos hay?

**Luego de recorrer esta parte del camino, entendemos mejor qué relación tiene la Ingeniería de Software con el área de conocimiento de Requisitos y cómo pueden los Requisitos ayudar a que una solución aporte valor en el negocio.**

## REFERENCIAS

---



Para profundizar:

- <https://www.computer.org/education/bodies-of-knowledge/software-engineering>
- <https://www.iiba.org/standards-and-resources/babok/>

- <https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/>

## PDF DESCARGABLE

---



SI397\_UN1\_LMM\_DESC\_SOFTWARE ENGINEERING\_V1.PDF

10.2 MB



Material producido por la Universidad Peruana de Ciencias Aplicadas

Autor: Ángel Augusto Vasquez Nuñez

SALIR

UPC, 2021.