



# Ingeniería de Software I



Profesora: Trinidad  
Latorre

**Comunicación:**

- [Trinidad.latorre@docentes.unpaz.edu.ar](mailto:Trinidad.latorre@docentes.unpaz.edu.ar)
- Campus



# Ingeniería de Software I



# CRONOGRAMA DE CLASES

Semana	Fecha	Tema
1	12/8/2025	PARO DOCENTE
2	19/8/2025	U1 – Procesos
3	26/8/2025	U1 – Procesos
4	2/9/2025	U2 - Sistemas de Información
5	9/9/2025	U2 - Sistemas de Información
6	16/9/2025	U3 – Requerimientos
<b>7</b>	<b>23/9/2025</b>	<b>Primer Examen Parcial</b>
8	30/9/2025	U3 – Requerimientos
<b>9</b>	<b>7/10/2025</b>	<b>Recuperatorio Primer Parcial</b>
10	17/10/2025	U3 – Requerimientos
11	21/10/2025	U3 – Requerimientos
12	28/10/2025	U4 – Calidad
13	4/11/2025	U4 – Calidad
<b>14</b>	<b>11/11/2025</b>	<b>Segundo Examen Parcial</b>
<b>15</b>	<b>18/11/2025</b>	<b>Recuperatorio Primer Parcial</b>
16	25/11/2025	Cierre
	2/12/2025	Examen Integrador
	15-20/dic '25	Examen Final

# Software

Que es?



# Software

## SOFTWARE



Es el equipamiento lógico o soporte lógico de una computadora.



Comprende el conjunto de los componentes necesarios que hacen posible la realización de tareas específicas





# ¿Qué es el software?

- **La suma total de** los programas de cómputo, procedimientos, reglas de documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo [[IEEE Computer Society Press, 1993](#)].
- **Es un producto** que diseñan y construyen los ingenieros de software. Esto abarca **programas** que se ejecutan dentro de una computadora de cualquier tamaño y arquitectura, **documentos** que comprenden formularios virtuales e impresos y **datos** que combinan números y texto y también incluyen representaciones de la información de audio, vídeo e imágenes [[Pressman, 2002](#)].

# Historia y evolución del software

## Primera Era 1950-1965

Se trabajaba con la idea de codificar y corregir

No existía una planificación previa.

No existía documentación de ningún tipo.

Existencia de pocos métodos formales.

Desarrollos a base de prueba y error.

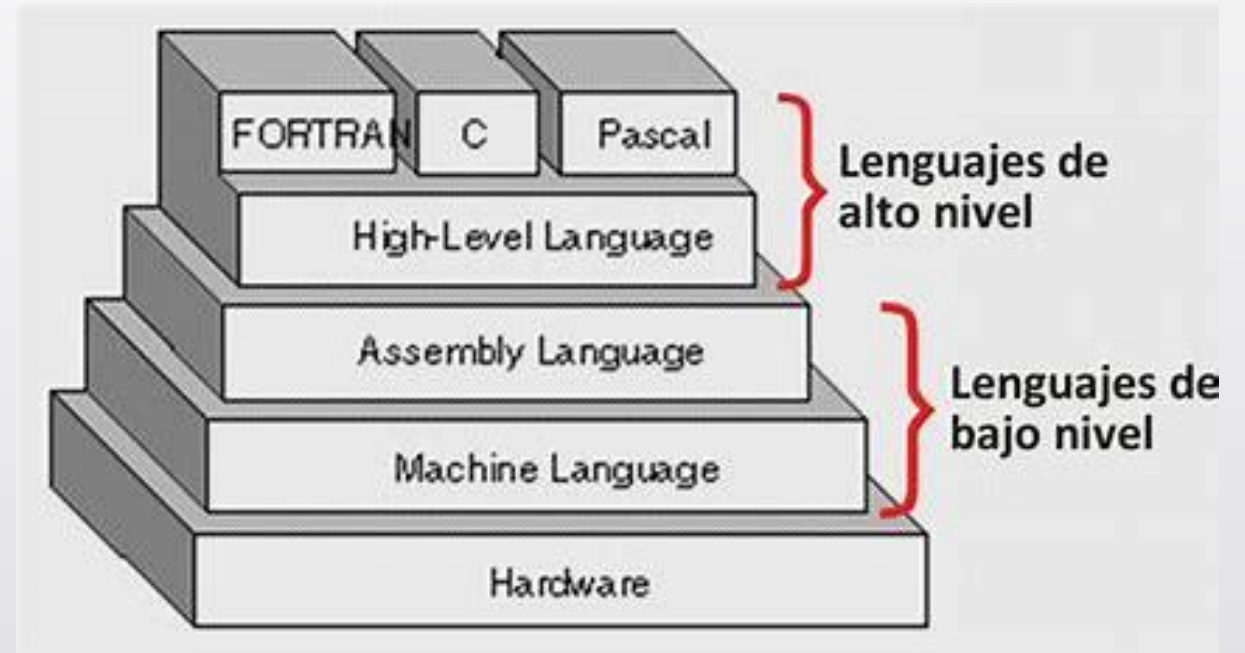




# Historia y evolución del software

## Segunda Era 1965-1972

Se buscaba simplificar código.  
Aparición de Multiprogramación y Sistemas Multisuarios. Sistemas de Tiempo Real apoyan la toma de decisiones.  
Aparición de software como producto.  
Se buscan procedimientos para el desarrollo de software.



# Historia y evolución del software



Tercera Era 1972 - 1985

Impacto colectivo de software.  
Aparecen las redes de información, tecnologías orientadas a objetos.  
Aparecen las redes Neuronales, sistemas expertos, SW de inteligencia artificial.  
La información como valor preponderante dentro de las organizaciones.

## Cuarta Era 1985-1995

Impacto colectivo de software  
Aparecen las redes de información, tecnologías orientadas a objetos  
Aparecen las redes Neuronales, sistemas expertos, SW de inteligencia artificial.  
La información como valor preponderante dentro de las organizaciones

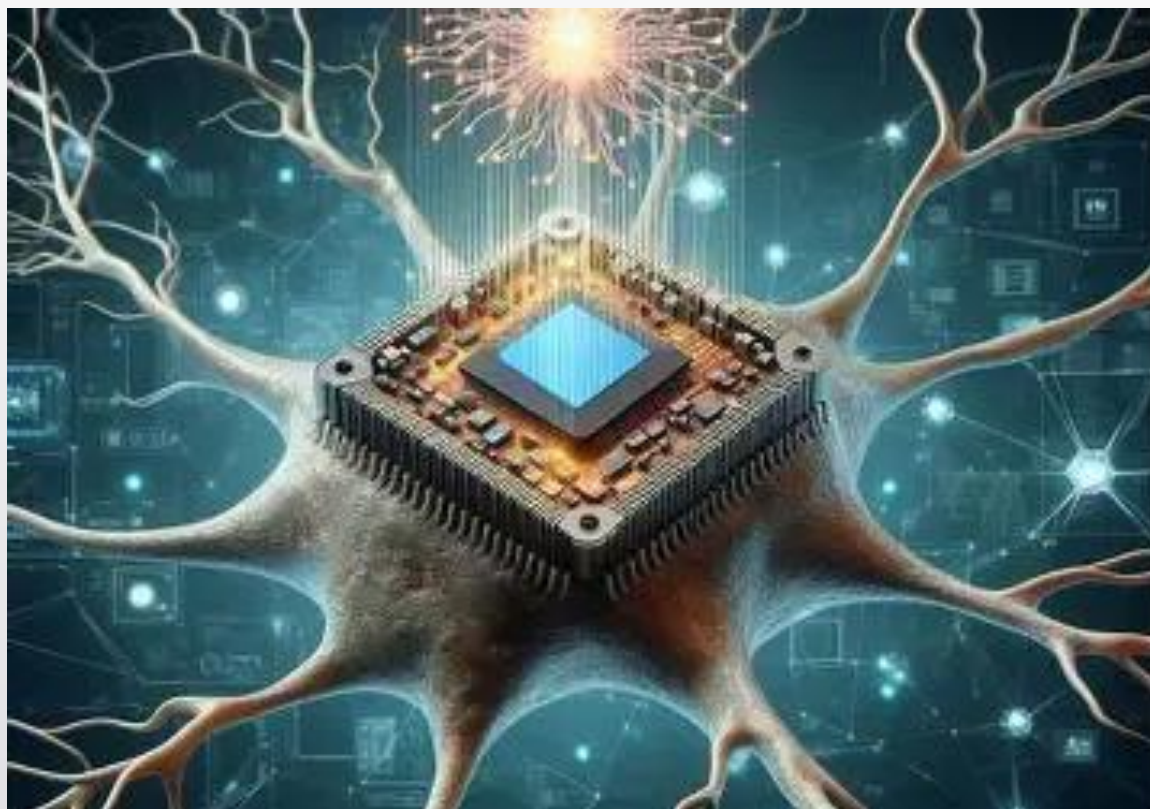




## Quinta Era del 2000

Utilizan algunos requisitos de las eras anteriores, aumenta la omnipresencia en la web. La reutilización y componentes de software.



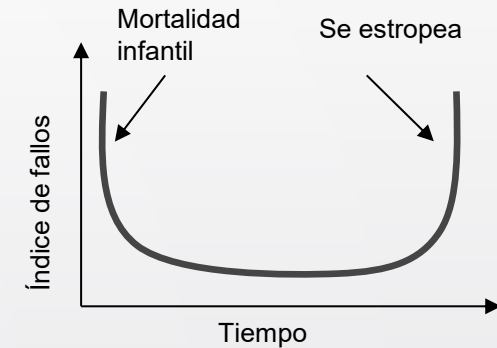


Actualidad

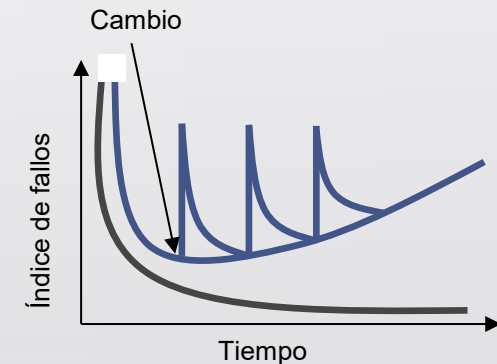


# Características del software

- El software al ser un elemento lógico tiene ciertas características que lo diferencian claramente respecto al hardware [Pressman, 2002].
  - El software se desarrolla, no se fabrica en un sentido clásico.
    - El desarrollo y fabricación generan un producto pero desde enfoques diferentes.
  - El software no se estropea; pero se deteriora.
    - Los fallos del hardware se dan al principio y al final de su vida, mientras que en el software el mantenimiento dado a lo largo de su vida introduce nuevos fallos.
  - Aunque la industria tiende a ensamblar componentes, la mayoría del software se construye a la medida.
    - Esta situación está cambiando con el uso más extendido de la programación orientada a objetos.



**Curva de fallos del hardware**



**Curvas de fallos real e idealizado del software**



# Dominios de aplicación del software

Actualmente hay siete categorías de software [[Pressman, 2010](#)]

## 1. Software de sistemas

- Conjunto de programas para servir a otros programas.
- En general tienen una fuerte interacción con el hardware, múltiples usuarios, operación concurrente, compartición de recursos, estructuras de datos complejas, entre otras.
- Ejemplos: compiladores, editores, utilidades de gestión de archivos, controladores, software de redes, etc.

## 2. Software de aplicación

- Programas aislados que resuelven una necesidad específica de negocios.
- Procesan datos comerciales o técnicos para facilitar las operaciones o toma de decisiones de negocios o técnicas.
- Ejemplos: procesamiento de transacciones en puntos de venta, control de procesos de manufactura en tiempo real.

## 3. Software de ingeniería y ciencias

- Se ha caracterizado por “algoritmos devoradores de números”.
- Las aplicaciones van desde la astronomía a la vulcanología, del análisis de tensiones en automóviles a la dinámica orbital de un transbordador espacial, de la biología molecular a la manufactura automatizada.



## **Dominios de aplicación del software (2)**

4. Software incrustado
  - Reside dentro de un producto o sistema y se usa para implementar y controlar funciones para el usuario final y para el sistema en sí.
  - Ejecuta funciones limitadas y particulares o provee una capacidad de funcionamiento y control.
  - Ejemplos: control del tablero de un microondas, funciones digitales en un automóvil como el control de combustible.
5. Software de línea de productos
  - Es diseñado para proporcionar una capacidad específica para uso de muchos consumidores diferentes.
  - Se centra en un mercado particular o a mercados masivos.
  - Ejemplos: control de inventario de productos, procesadores de textos, hoja de cálculo, etc.



## **Dominios de aplicación del software (3)**

### **6. Aplicaciones Web**

- Llamadas Webapps. Esta categoría de software centrado en redes agrupa una amplia gama de aplicaciones.
- Van desde sencillas páginas dinámicas hasta ambientes de cómputo sofisticados con integración a bases de datos corporativas y aplicaciones de negocios.

### **7. Software de inteligencia artificial**

- Hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados los análisis directos.
- Ejemplos: sistemas expertos o basados en conocimientos, reconocimiento de patrones, imágenes, voz, redes neuronales artificiales, etc.



# La crisis del software

- La mayoría de los expertos están de acuerdo en que la manera más probable para que el mundo se destruya es por accidente. Ahí es donde nosotros entramos; somos profesionales de la informática, provocamos accidentes. [[Nathaniel Borenstein](#)].
- Este problema se analizó en 1968 denominándose “**crisis del software**” y provocó que **muchos proyectos de software fracasarán o nunca se terminarán**.





# La crisis del software

El término de “crisis del software” empezó a ser usado en 1968 en una conferencia organizada por la OTAN, donde se abordó de forma directa el concepto. En dicha conferencia se pretendía explicar la serie de problemas que estaban enfrentando los proyectos de software, como por ejemplo retrasos y altos costos que diferían del presupuesto previsto al inicio del proyecto (Sommerville, 2005).

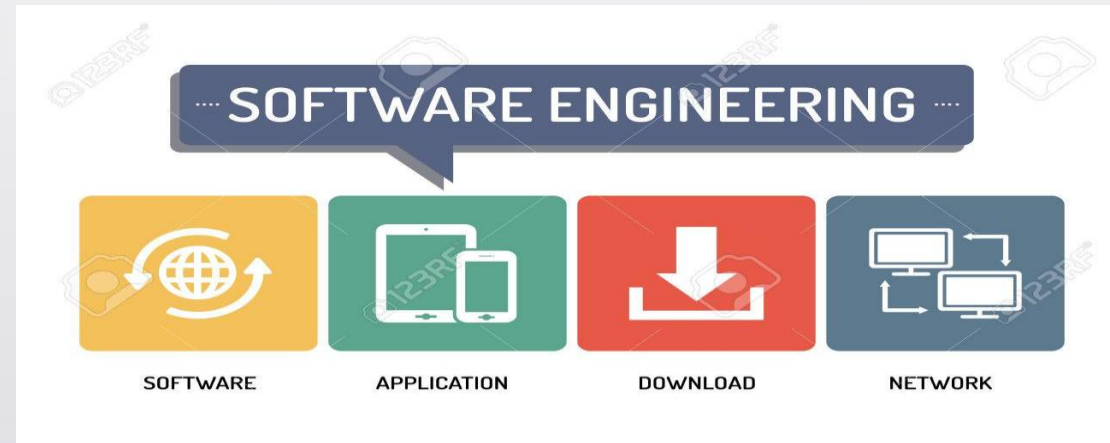


Por otra parte, también estaba en cuestionamiento la calidad de los productos desarrollados que apenas alcanzaban un nivel aceptable.



A partir de esa conferencia surge la idea de considerar el software como un producto y que, al igual que otros productos, se debe someter a un proceso de fabricación

# Ingeniería de Software





# Evolución de la Ingeniería de Software

- En respuesta a esta “crisis” se comenzó a **buscar los factores provocan los problemas de calidad** y conocer cuáles eran los **procesos necesarios para crear el software y mantenerlo funcionando** llegando a determinarse que era necesario **profesionalizar el desarrollo de software** y considerar aspectos básicos para **crear un producto de calidad acorde a las necesidades del cliente.**



## Evolución de la Ingeniería de Software (2)

*De acuerdo a Pressman (2006), a partir de ese momento la comunidad asociada al desarrollo de software (universidades y organismos de estandarización) comenzaron a dar forma a **la idea de la ingeniería del software**, proponiendo **una disciplina** que permitiera ver la creación de software como un proceso de producción sistematizado que facilitara la entrega de productos de calidad y a tiempo.*

# ¿Qué es la Ingeniería de Software?

Es una de las ramas de las ciencias de la computación que estudia la creación de software confiable y de calidad, basándose en métodos y técnicas de ingeniería





# Ingeniería de Software

Existen varias definiciones para Ingeniería de Software, de acuerdo con los principales autores de esta disciplina, sin embargo, una de las más aceptadas es la propuesta por la organización IEEE en 1993:

**“La Ingeniería de Software es la aplicación de métodos sistemáticos, disciplinados y cuantificables para el desarrollo, operación y mantenimiento de software” [IEEE93].**



# Ingeniería de Software - Definiciones

- La ingeniería de software es el establecimiento y uso de principios robustos de la ingeniería a fin de obtener económicamente software que sea fiable y que funcione eficientemente sobre máquinas reales [[Bauer, 1972](#)].
- Ingeniería del software es la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadora y la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como desarrollo de software o producción de software [[Bohem, 1976](#)].
- La ingeniería de software es el estudio de los principios y metodologías para desarrollo y mantenimiento de sistemas de software [[Zelkovitz, 1978](#)].
- La Ingeniería de Software es una disciplina de la Ingeniería que concierne a todos los aspectos de la producción de software [[Sommerville, 1995](#)].

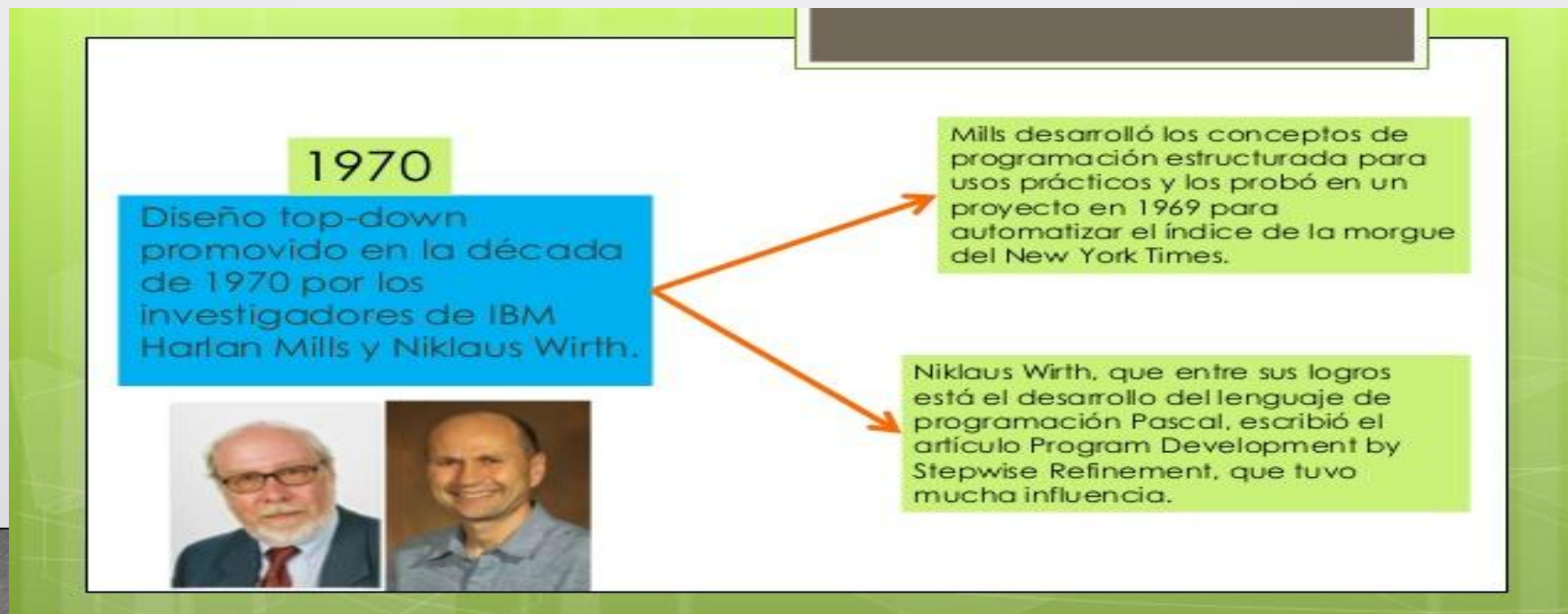
# ¿Qué propone la Ingeniería de software?

- Definir las **áreas de conocimiento** para el desarrollo de software.
- Definir con claridad los **procesos** que intervienen en el **desarrollo, mantenimiento y operación** del software.
- Extraer modelos a partir de las **mejores prácticas** de la industria.
- Definir **criterios unificados** para las diversas tareas involucradas en el software, por ejemplo, **estándares** o **recomendaciones técnicas**.



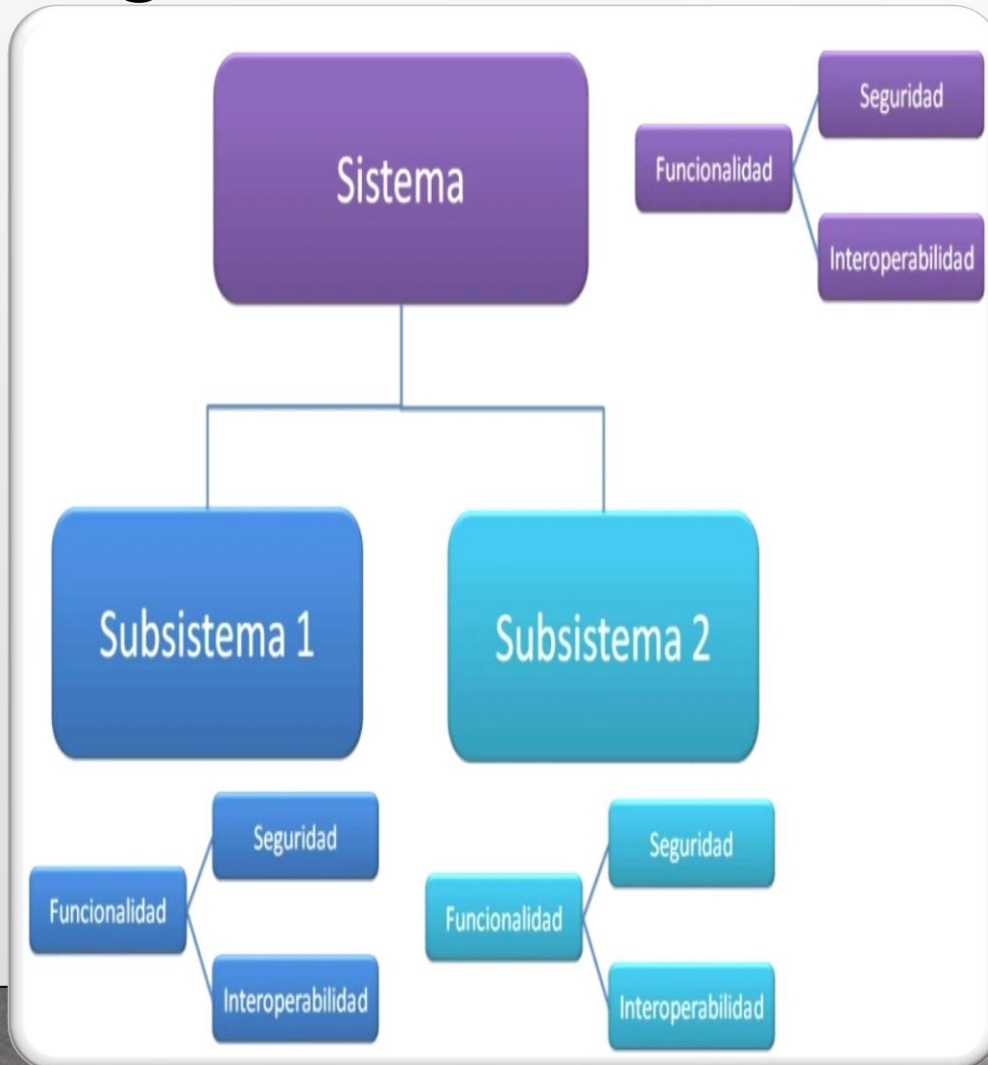
# Ingeniería de Software... un poco de historia

- Los primeros pasos orientados al diseño de software los realizó IBM, con Harlan Mills y Niklaus Wirth, quienes propusieron en los años 70 el Top Down Design, un diseño que permite descomponer la aplicación en una serie de módulos y funcionalidades, partiendo de las características globales y descendiendo a lo detallado y funcional mediante la creación de módulos, subprogramas, funciones y subrutinas, pero siempre cobijados bajo la programación estructurada.





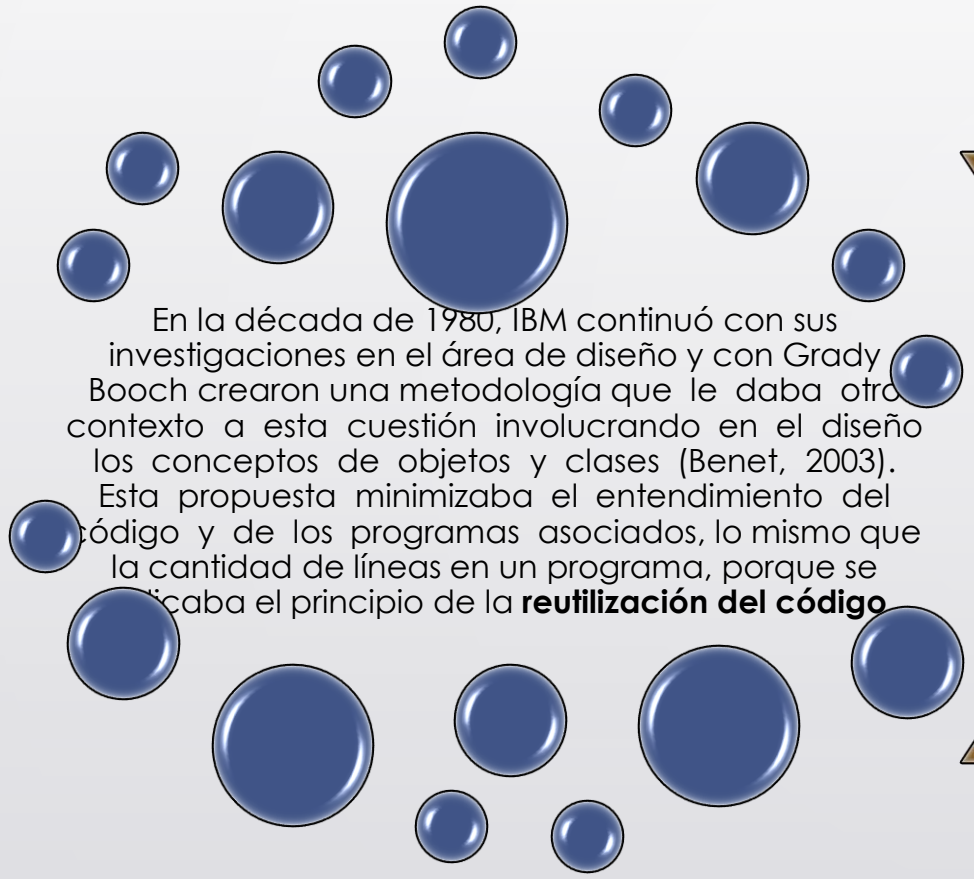
# Ingeniería de Software... un poco de historia



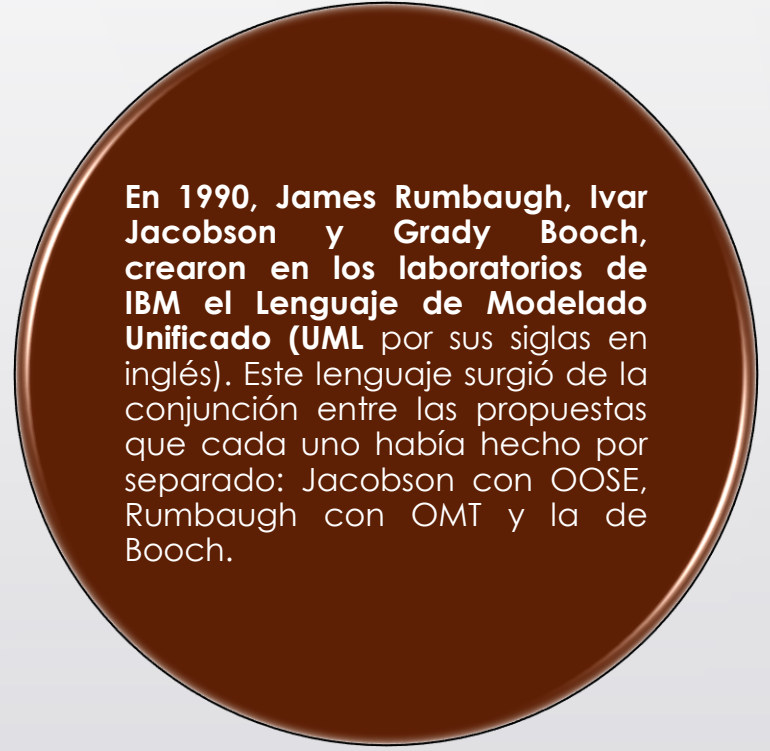
- El diseño top-down es una herramienta que presenta en primer lugar una solución a un problema general utilizando tres o cuatro pasos solamente. Cada uno de esos pasos en la primera solución se dividen en otros subpasos. Este proceso se repite varias veces, en cada iteración se produce una solución más detallada al problema original. Cuando los pasos ya no se pueden subdividir, el algoritmo ha terminado.
- **El diseño top-down también se conoce como descomposición funcional o refinamiento de pasos.**



# Ingeniería de Software... un poco de historia



En la década de 1980, IBM continuó con sus investigaciones en el área de diseño y con Grady Booch crearon una metodología que le daba otro contexto a esta cuestión involucrando en el diseño los conceptos de objetos y clases (Benet, 2003). Esta propuesta minimizaba el entendimiento del código y de los programas asociados, lo mismo que la cantidad de líneas en un programa, porque se aplicaba el principio de la **reutilización del código**.



En 1990, James Rumbaugh, Ivar Jacobson y Grady Booch, crearon en los laboratorios de IBM el **Lenguaje de Modelado Unificado (UML)** por sus siglas en inglés). Este lenguaje surgió de la conjunción entre las propuestas que cada uno había hecho por separado: Jacobson con OOSE, Rumbaugh con OMT y la de Booch.

////////////////////////////////////

También, incorpora el análisis precedente de la situación, el bosquejo del proyecto, el desarrollo del software, el ensayo necesario para comprobar su funcionamiento correcto y poner en funcionamiento el sistema.

Desarrollo del  
software

=

Ciclo de vida del  
software





# Ciclo de vida del software

- Cuando no se sigue un ciclo de vida y apenas se planea, se tiende a seguir el enfoque de “codificar y probar” lo que genera: una alta probabilidad de falla en el software, poca flexibilidad para modificaciones, no satisfacer plenamente los requisitos y descontento de los clientes [[Piatfina](#)].
- **Qué es un ciclo de vida?:**
  - Un modelo de ciclo de vida es un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso [[ISO/IEC 12207-1](#)].
  - Ciclo de vida del software es una aproximación lógica a la adquisición, suministro, el desarrollo, la explotación y el mantenimiento del software [[IEEE 1074](#)].

# Ciclo de vida del software



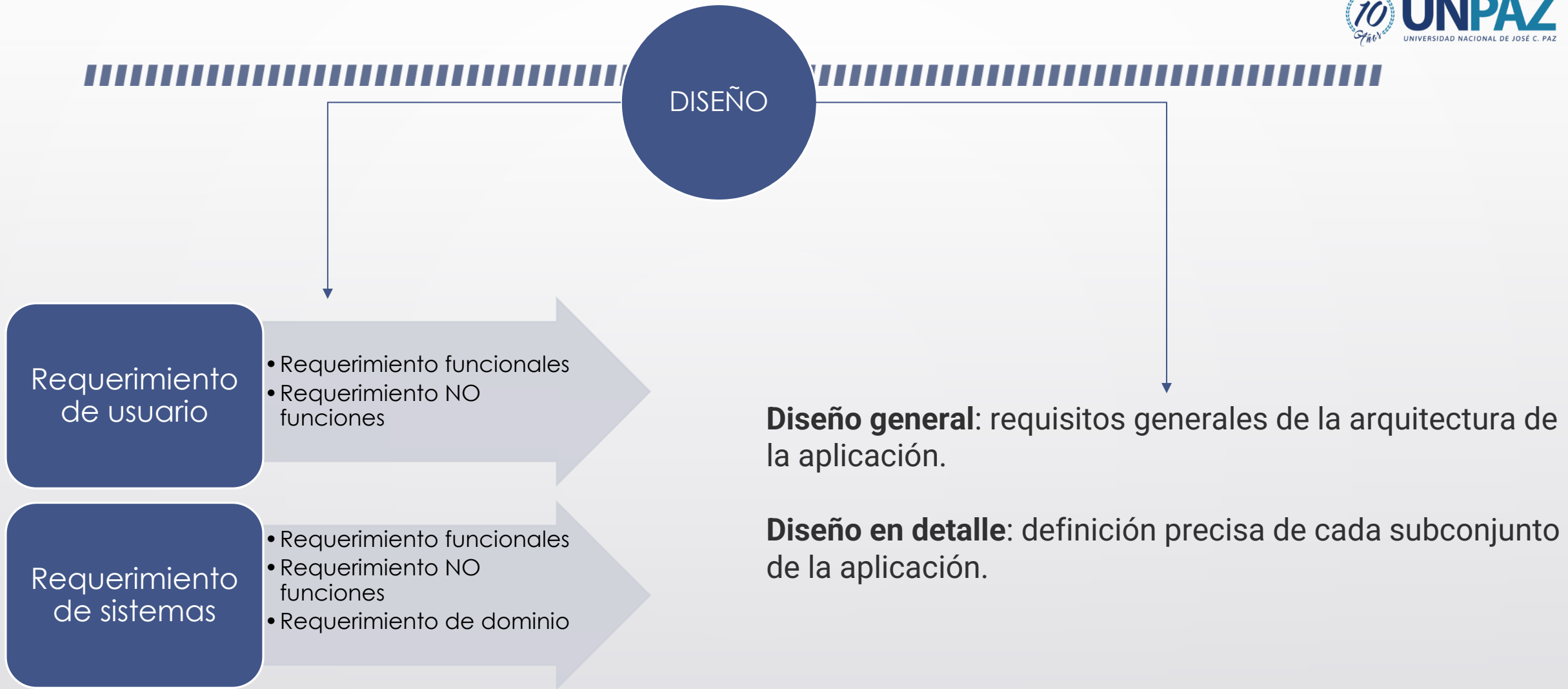
## ANÁLISIS



**Definición de objetivos:** define la finalidad del proyecto y su papel en la estrategia global.

**Análisis de los requisitos y su viabilidad:** recopila, examina y formula los requisitos del cliente y examina cualquier restricción que se pueda aplicar.





DESA-  
RROLLO

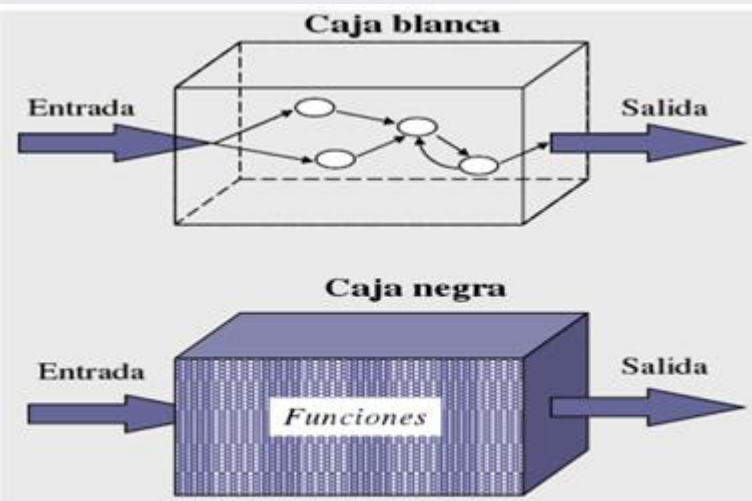


**Programación** (programación e implementación):  
implementación de un lenguaje de programación para  
crear las funciones definidas durante la etapa de diseño.

## PRUEBAS

**Prueba de unidad:** prueba individual de cada subconjunto de la aplicación para garantizar que se implementaron de acuerdo con las especificaciones.

**Los Métodos de Prueba de Software** tienen el objetivo de diseñar pruebas que descubran diferentes tipos de errores con menor tiempo y esfuerzo.



a. Técnicas de Diseño de Casos de Pruebas

Las pruebas de **caja blanca** se basan en un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, comprobando las condiciones y ciclos; examinando el estado del programa en varios puntos. (Criterios basados en el contenido de los módulos).

Las pruebas de **caja negra** se aplican a la interfaz del software, examinando el aspecto funcional del sistema. Pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. (Criterios basados en las interfaces y las especificaciones de los módulos).

## IMPLEMENTACIÓN

El despliegue comienza cuando el código ha sido suficientemente probado, ha sido aprobado para su liberación y ha sido distribuido en el entorno de producción.

SERVIDOR QA

SERVIDOR TESTING

SERVIDOR PRODUCTIVO

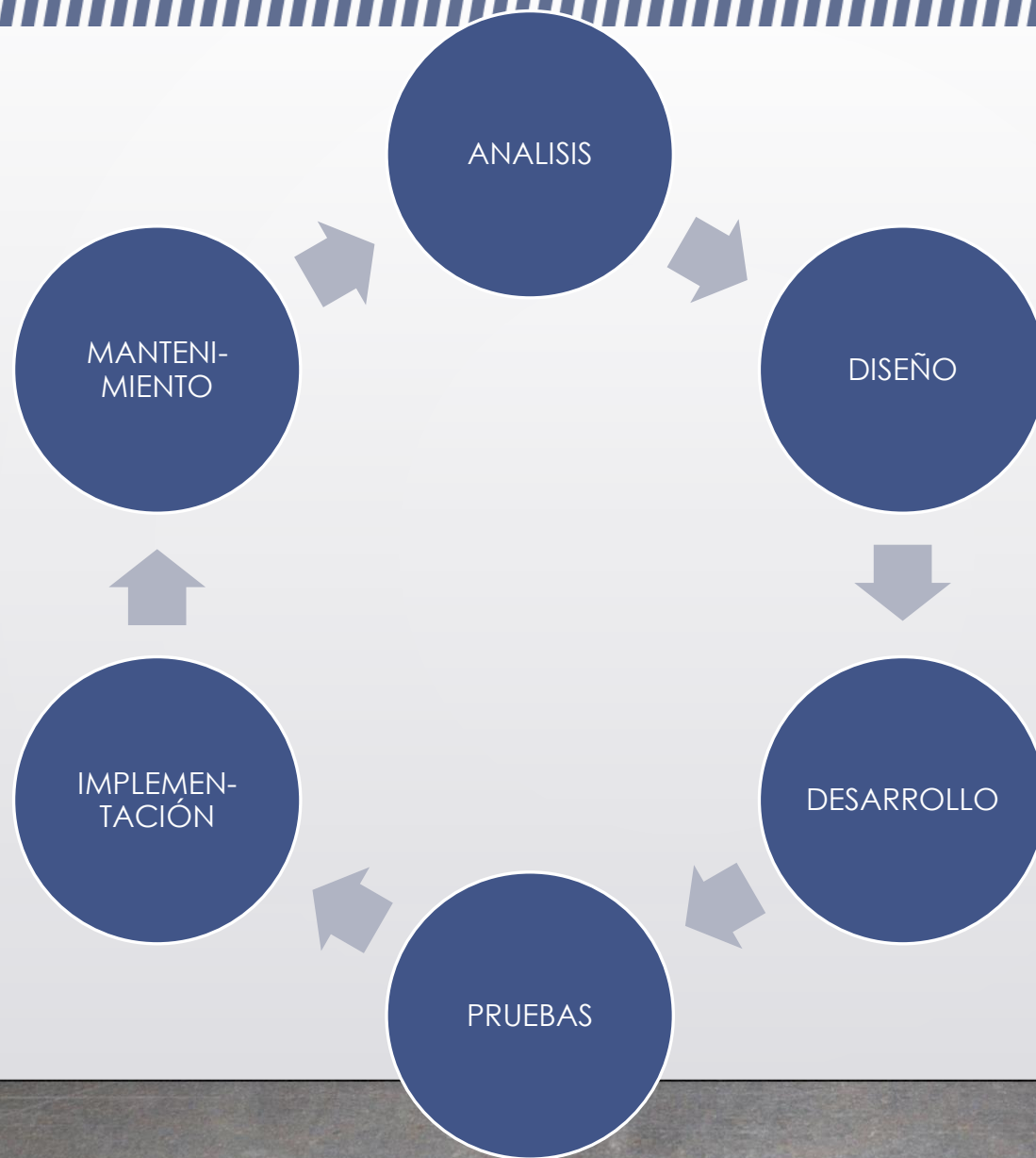




## MANTENIMIENTO

El **mantenimiento de software** es la modificación de un producto de software después de la entrega, para corregir errores, mejorar el rendimiento, u otros atributos.

El mantenimiento del software es una de las actividades más comunes en la ingeniería de software.



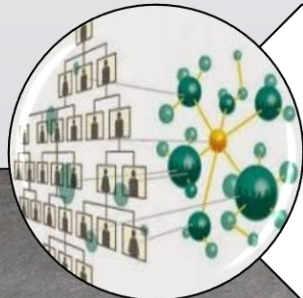
# INGENIERIA DEL SOFTWARE ACTUALMENTE



En la actualidad el software tiene un doble papel. Es el producto, pero al mismo tiempo, actúa como el conductor que entrega el producto. Como conductor utilizado para entregar el producto, actúa como base de control, por ejemplo un sistema operativo, o un sistema gestor de redes.

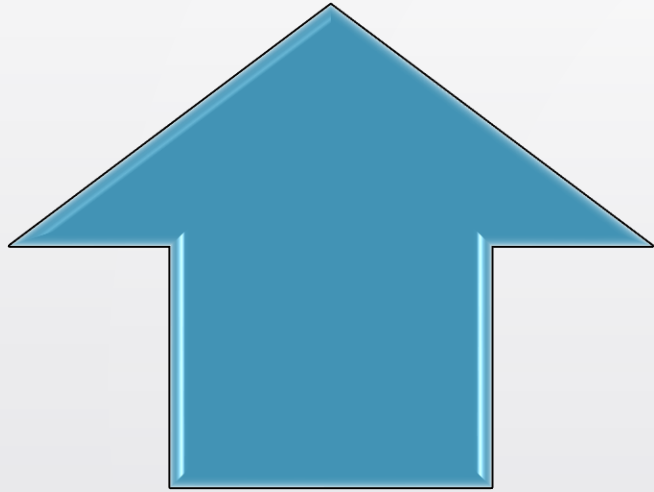


El software actúa como distribuidor y hace llegar a los usuarios, el producto más importante de este siglo: la información.

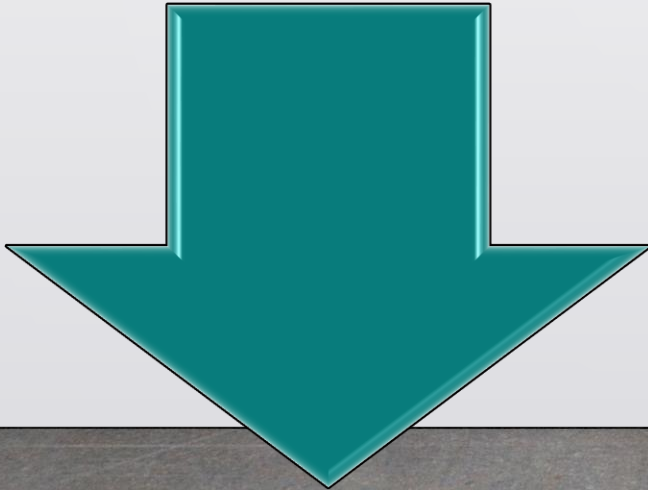


El software modifica la información personal para hacerlos más útiles en un entorno local, administra información comercial para mejorar la competitividad, facilita el acceso a redes a nivel mundial, y propone la forma de obtener información de cualquier manera.

# INGENIERIA DEL SOFTWARE ACTUALMENTE



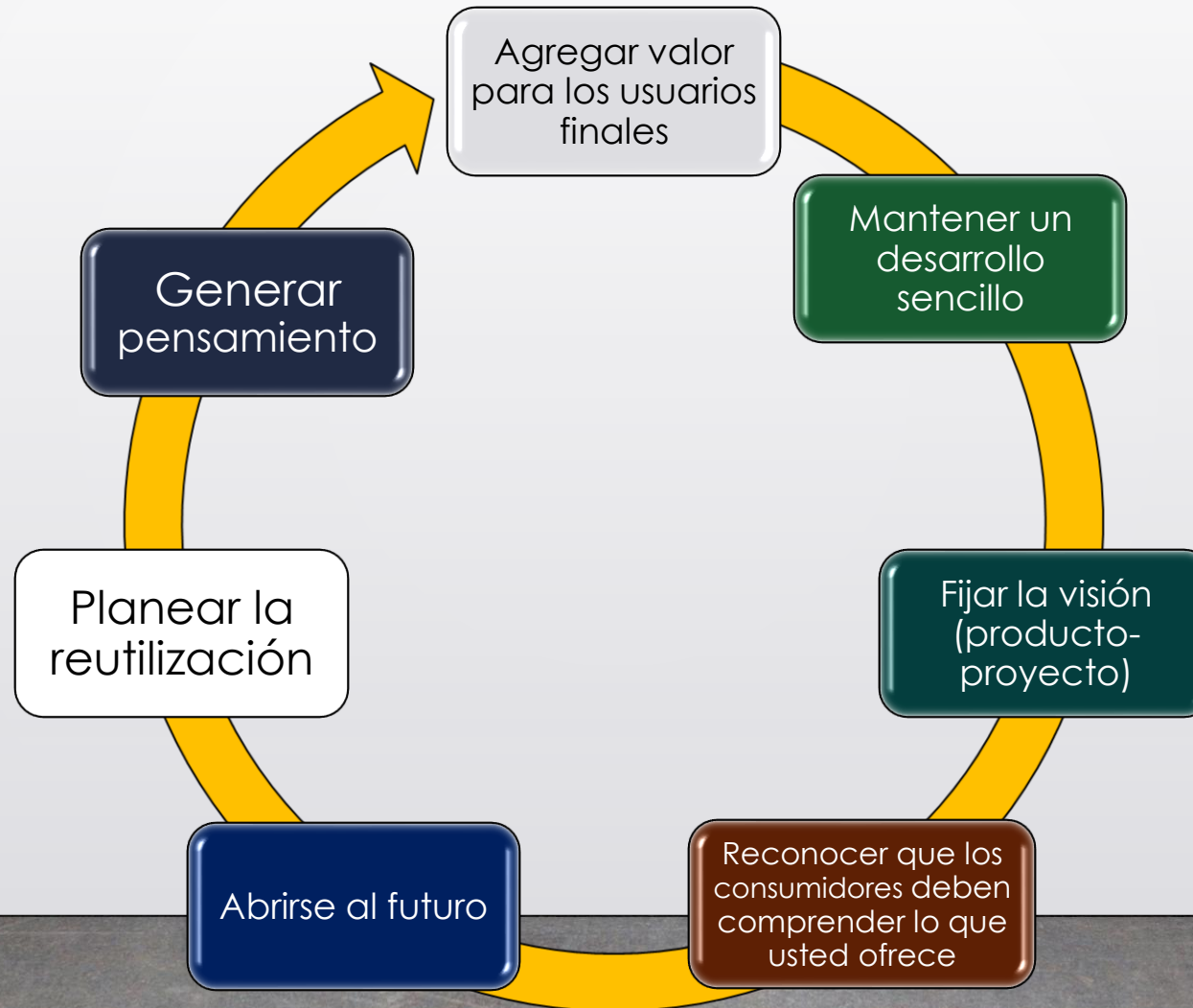
En la actualidad la ingeniería del Software es considerada una nueva especialidad de la ingeniería y junto con la especialidad de Ingeniería Informática, es una de las profesiones con más demanda, aunque hay lugares en el mundo, en los que no es muy bien remunerada



La ingeniería del software trata campos muy variados de la informática y de las Ciencias de la Computación, que además se aplican a un amplio espectro de campos, tales como negocios, investigación científica, medicina, producción, logística, banca, meteorología, derecho, redes, entre otras muchas



# PRINCIPIOS DE LA INGENIERIA DEL SOFTWARE



A decorative horizontal line consisting of many small, slanted blue dashes spans the width of the slide, positioned just below the header.

¿Cuál es la etapa más importante?





## ¿ Por qué formar equipo en el desarrollo de software?

- Debido a su complejidad
- A la necesidad de personas con distintas habilidades
- El aporte de todas las capacidades necesarias dentro de un equipo llevara al cumplimiento del objetivo
- Es posible que no se requieran todos los roles en un desarrollo, eso dependerá del tamaño y el tipo de desarrollo



# ROLES



Administrador de proyectos



Analista



Diseñador



Programador



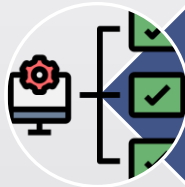
Téster



Documentador



Ingeniero de validación



Administrador de configuraciones



Cliente



CRM



La solicitud del usuario



Lo que entendió el líder del proyecto



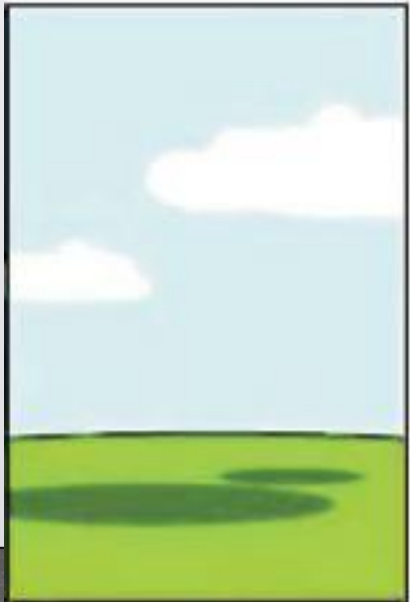
El diseño del analista de sistemas



El enfoque del programador



La recomendación del consultor externo



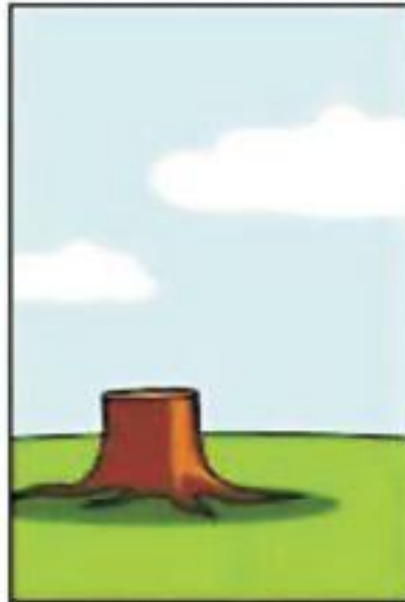
La documentación del proyecto



La implantación en producción



El presupuesto del proyecto



El soporte operativo



Lo que el usuario realmente necesitaba



# Roles desarrollo del software



¿ Qué tiene que ver esa fabula ?