# DESARROLLO DE SOFTWARE

Clase I

# DEFINICIONES

## Software

Programa de computadora y la documentación asociada. Los productos de software se pueden desarrollar para algún cliente en particular o para un mercado en general (genéricos).

## Ing. en Software

Comprende todos los aspectos de la producción de Software. *Buenas Prácticas*

# DIFERENCIAS

## Ingeniería en Software

Prácticas para desarrollar y liberar un software útil.

## Ciencias de la Computación

Teoría y fundamentos referentes a las computadoras y los sistemas de Software.

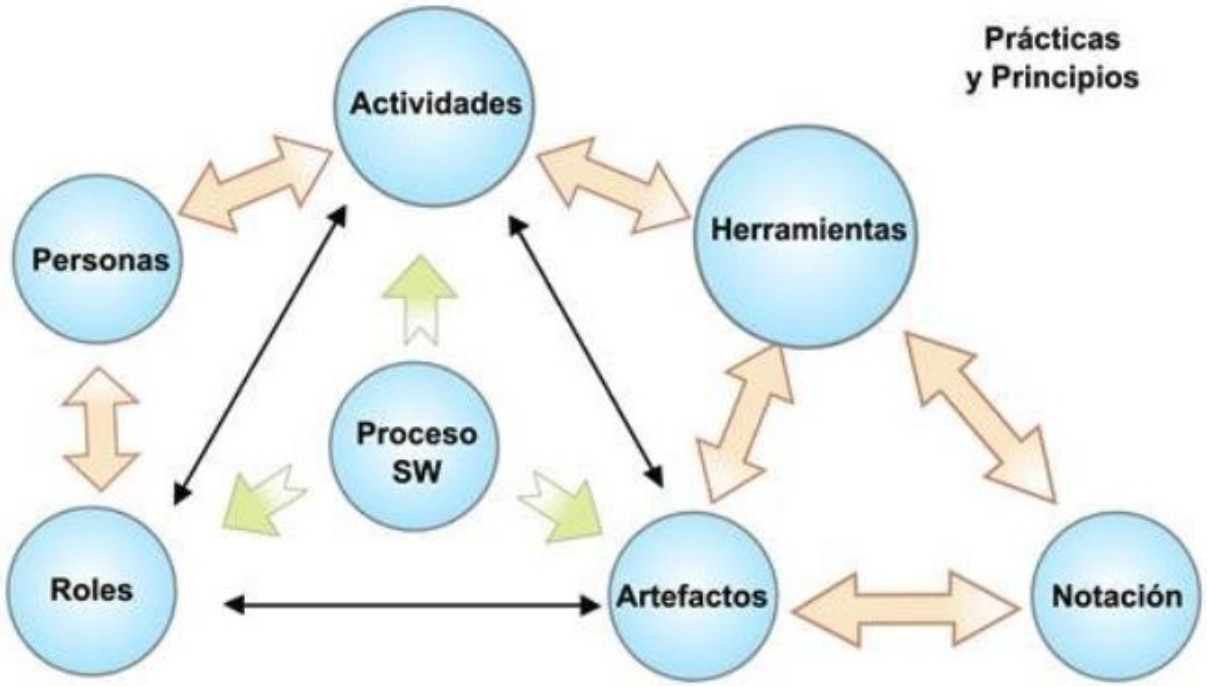
## Ingeniería en Sistemas

Todos los aspectos del desarrollo de un sistema, el software es parte del mismo.

# PROCESO DE SOFTWARE

Actividades que sirven para desarrollo, evolución o mantenimiento de un software.

Hay modelos que se adaptan, dependiendo del tipo d software / proyecto a realizar.



# COSTOS

**Nuevo SW**

20% dirección
40% desarrollo
30% pruebas
10% despliegue

**Depende el software:**

20% dirección
30% Integración
40% pruebas
10% despliegue

**Legacy**

20% dirección
20% desarrollo
50% pruebas
10% despliegue

**Evolución**

# QUE ES UN BUEN SOFTWARE ?

## Mantenible

Tiene que poder evolucionar por cualquier cambio de requerimientos.

## Confiabilidad

Tiene que ser seguro, cualquier falla tiene que ser tratada e informada.

## Eficiencia

No tenemos que necesitar un reactor nuclear para que nuestro software funcione

## Usabilidad

Se tiene que entender el funcionamiento, sencillo, simple, apb

# RETOS DEL SOFTWARE

**Heredado:**
A mayor cantidad de años mayor cantidad de sistemas heredados (legacy)



**Heterogeneidad:**
Nuevos SW y herramientas para conectarse

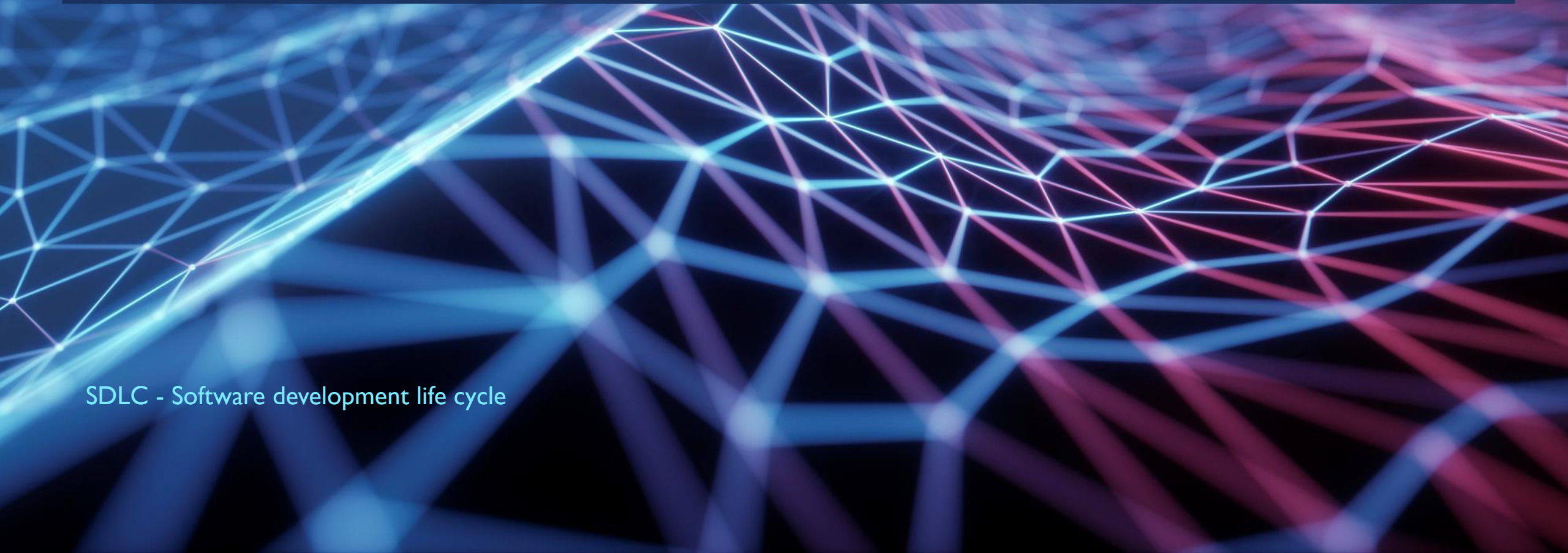
**Tiempo:**
El tiempo es crítico para que un producto de SW funcione en el mercado con las expectativas correspondientes.

# PROBLEMAS DEL SOFTWARE

- La planificación y estimación de costos frecuentemente son imprecisas.
  - Falta de productividad.
  - La calidad del software es a veces inaceptable.
- Estos problemas al final crean insatisfacción y falta de confianza de los clientes. Los problemas anteriores son sólo manifestación de otras dificultades:
  - No tenemos tiempo de recoger datos sobre el proceso de desarrollo del software.
  - Los proyectos de desarrollo de software se llevan a cabo con sólo una vaga indicación de los requisitos del cliente.
  - El mantenimiento de software es muy costoso y no se le ha considerado un aspecto importante.

"*Los problemas anteriores son solucionables, dándoles un enfoque de ingeniería al desarrollo de software*"

# Ciclo de Vida del Desarrollo de SW

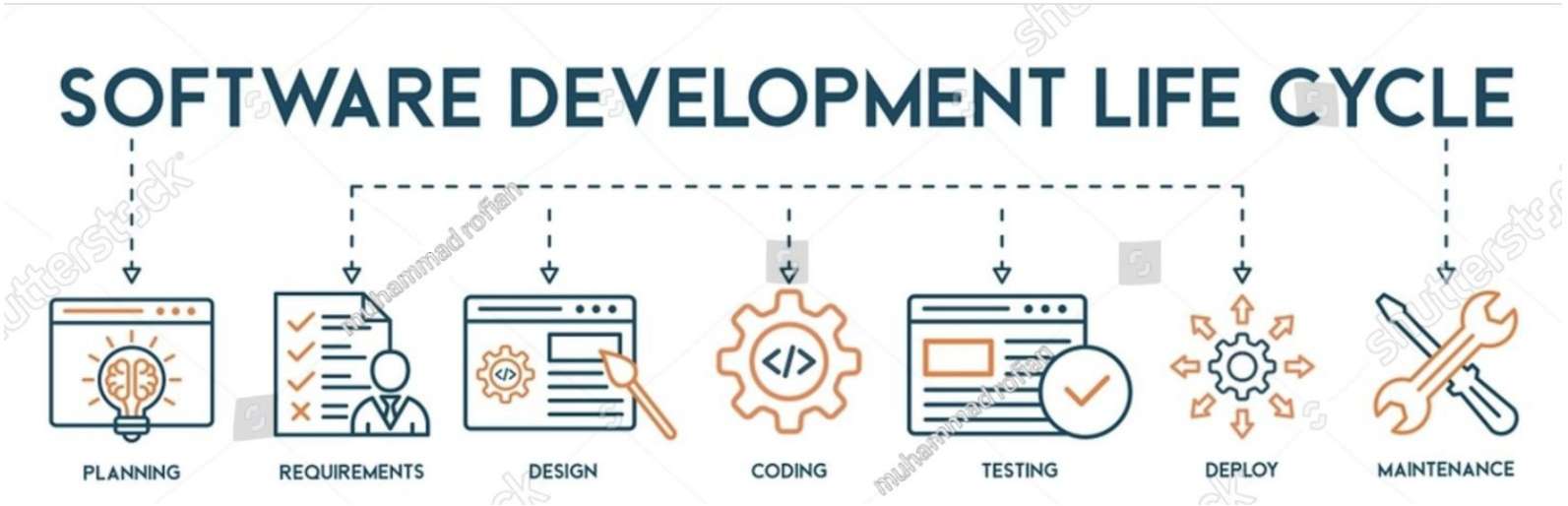


SDLC - Software development life cycle

# Metodologías y Ciclo de Vida de Desarrollo de Software

| Característica | SDLC | Metodologías de desarrollo de software |
| --- | --- | --- |
| Definición | Un marco general que describe las etapas involucradas en el desarrollo de software. | Conjuntos específicos de prácticas y herramientas que se utilizan para desarrollar software. |
| Enfoque | Lineal o iterativo | Varía según la metodología |
| Ejemplos | Cascada, ágil, espiral | Scrum, Kanban, XP |

# FASES DE DESARROLLO




- Análisis: Gestión de Requerimientos
- Diseno: Arquitectura
- Implementación: Desarrollo
- Pruebas: Testing
- Mantenimiento

# SDLC TRADICIONALES

- CASCADA O WATERFALL
- PROTOTIPADO O PROTOTYPING
- ESPIRAL O SPIRAL
- INCREMENTAL
- MODELO “V”

# CASCADA


Proceso de desarrollo
Análisis
Comprobación
Diseño
Documento de diseño
Comprobación
Implementación
Arquitectura de software
Comprobación
Verificación
Software
Mantenimiento

# CASCADA

- **Ingeniería y Análisis del Sistema**: debido que el software es siempre parte de un sistema mayor, el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requisitos al software.
- **Análisis de los requisitos del Software**: el proceso de recopilación de los requisitos se centra e intensifica especialmente en el software. El ingeniero de software (Analistas) debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridas.
- **Diseño**: el diseño del software se enfoca en cuatro atributos distintos del programa: la estructura de los datos, la arquitectura del software, el detalle procedimental (backend) y la caracterización de la interfaz (front end). El proceso de diseño traduce los requisitos en una representación del software técnica.

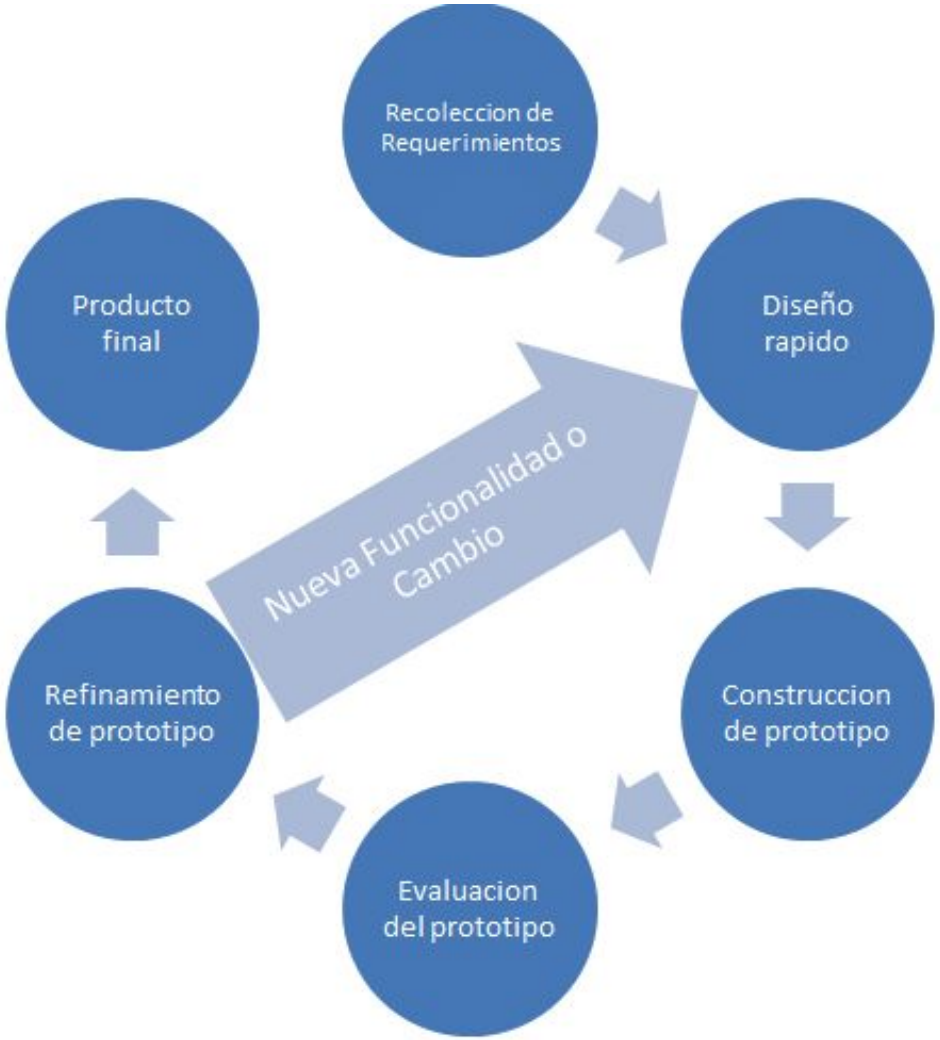
# CASCADA

- **Implementación (Desarrollo)**: el diseño debe traducirse en una forma legible para la máquina. El paso de codificación realiza esta tarea. Si el diseño se realiza de una manera detallada la codificación puede realizarse mecánicamente.
- **Verificación (QA)**: una vez que se ha generado el código comienza la prueba del programa. La prueba se centra en la lógica interna del software, y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.
- **Mantenimiento**: el software sufrirá cambios después de que se entrega al cliente. Los cambios ocurrirán debido a se han encontrado errores, a que el software deba adaptarse a cambios del entorno externo (sistema operativo o dispositivos periféricos), o debido a que el cliente requiera ampliaciones funcionales o del rendimiento.

# CASCADA

- **Desventajas:**
  - Los proyectos reales raramente siguen el flujo secuencial que propone el modelo, siempre hay iteraciones y se crean problemas en la aplicación del paradigma.
  - Normalmente, es difícil para el cliente establecer explícitamente al principio todos los requisitos. El ciclo de vida clásico lo requiere y tiene dificultades en acomodar posibles incertidumbres que pueden existir al comienzo de muchos productos.
  - El cliente debe tener paciencia. Hasta llegar a las etapas finales del proyecto, no estará disponible una versión operativa del programa. Un error importante no detectado hasta que el programa esté funcionando puede ser desastroso.

# PROTOTIPADO - POC



# PROTOTIPADO - POC

- Un prototipo es una versión preliminar de un sistema de información con fines de demostración o evaluación.
- El prototipo de requerimientos es la creación de una implementación parcial de un sistema, para el propósito explícito de aprender sobre los requerimientos del sistema. Un prototipo es construido de una manera rápida tal como sea posible.
- Esto es dado a los usuarios, clientes o representantes de ellos, posibilitando que ellos experimenten con el prototipo. Estos individuos luego proveen la retroalimentación sobre lo que a ellos les gustó y no les gustó acerca del prototipo proporcionado, quienes capturan en la documentación actual de la especificación de requerimientos la información entregada por los usuarios para el desarrollo del sistema real. El prototipo puede ser usado como parte de la fase de requerimientos (determinar requerimientos) o justo antes de la fase de requerimientos (como predecesor de requerimientos). En otro caso, el prototipo puede servir su papel inmediatamente antes de algún o todo el desarrollo incremental en modelos incremental o evolutivo.

# PROTOTIPADO

- El prototipo ha sido usado frecuentemente en los 90, porque la especificación de requerimientos para sistemas complejos tiende a ser relativamente dificultoso de cursar. Muchos usuarios y clientes encuentran que es mucho más fácil proveer retroalimentación convenientemente basada en la manipulación, leer una especificación de requerimientos potencialmente ambigua y extensa.
- Diferente del modelo evolutivo donde los requerimientos mejor entendidos están incorporados, un prototipo generalmente se construye con los requerimientos entendidos más pobremente.
- En caso que ustedes construyan requerimientos bien entendidos, el cliente podría responder con "sí, así es", y nada podría ser aprendido de la experiencia.

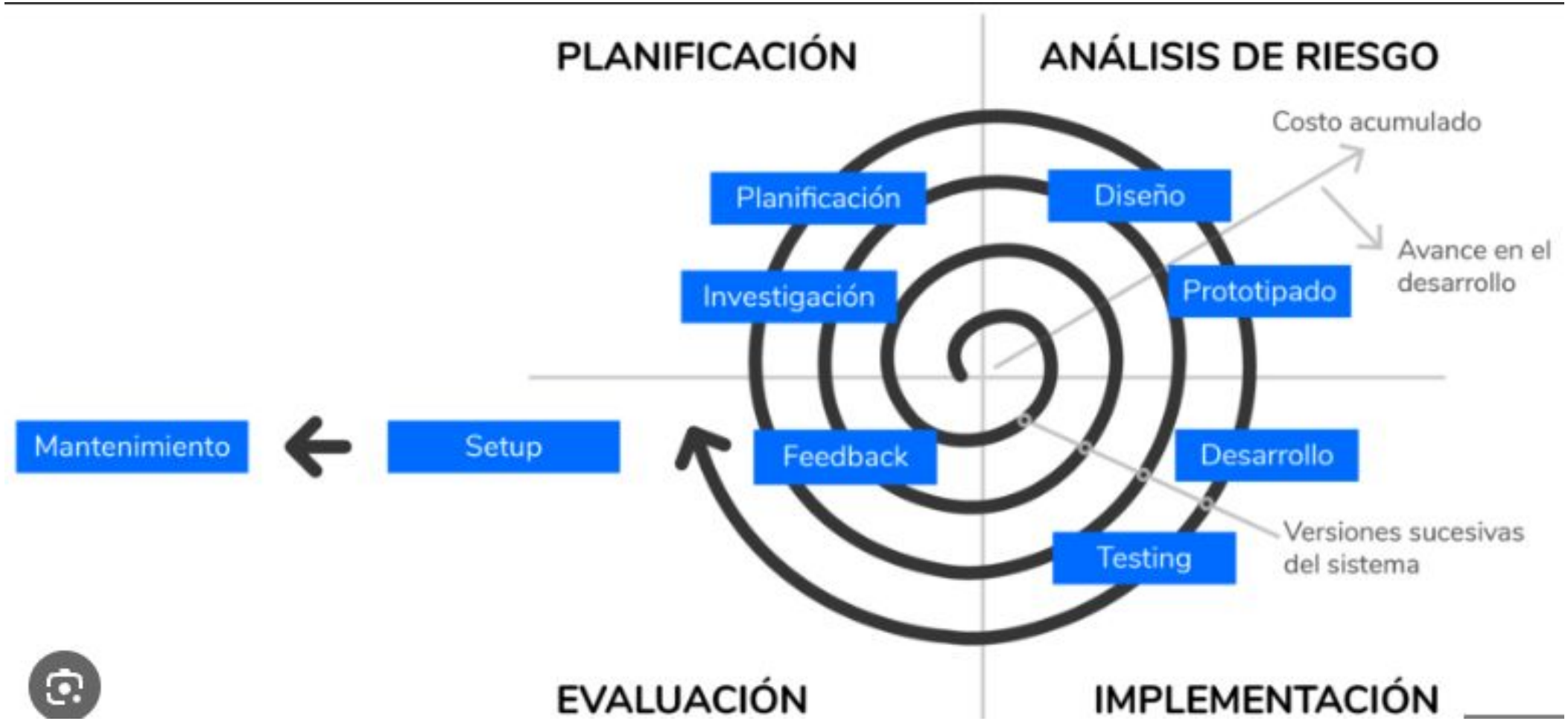
# PROTOTIPADO - POC

- **Ventajas**:
  - Útiles cuando los requerimientos son cambiantes.
  - Cuando no se conoce bien la aplicación.
  - Cuando el usuario no se quiere comprometer con los requerimientos.
  - Cuando se quiere probar una arquitectura o tecnología.
  - Cuando se requiere rapidez en el desarrollo.

# PROTOTIPADO

- **Desventajas**:
  - No se conoce cuándo se tendrá un producto aceptable.
  - No se sabe cuántas iteraciones serán necesarias.
  - Da una falsa ilusión al usuario sobre la velocidad del desarrollo, sin tener conocimiento que el prototipo es prueba, hay funciones que no se han tenido en cuenta, la calidad del software global o la facilidad de mantenimiento a largo plazo. Cuando se informa de que el producto se debe construir otra vez para que se puedan mantener los niveles altos de calidad, el cliente no lo entiende y pide que se apliquen pequeños ajustes para que se pueda hacer del prototipo un producto final.

# ESPIRAL


PLANIFICACIÓN
ANÁLISIS DE RIESGO
Costo acumulado
Planificación
Diseño
Avance en el desarrollo
Investigación
Prototipado
Mantenimiento
Setup
Feedback
Desarrollo
Versiones sucesivas del sistema
Testing
EVALUACIÓN
IMPLEMENTACIÓN

# ESPIRAL

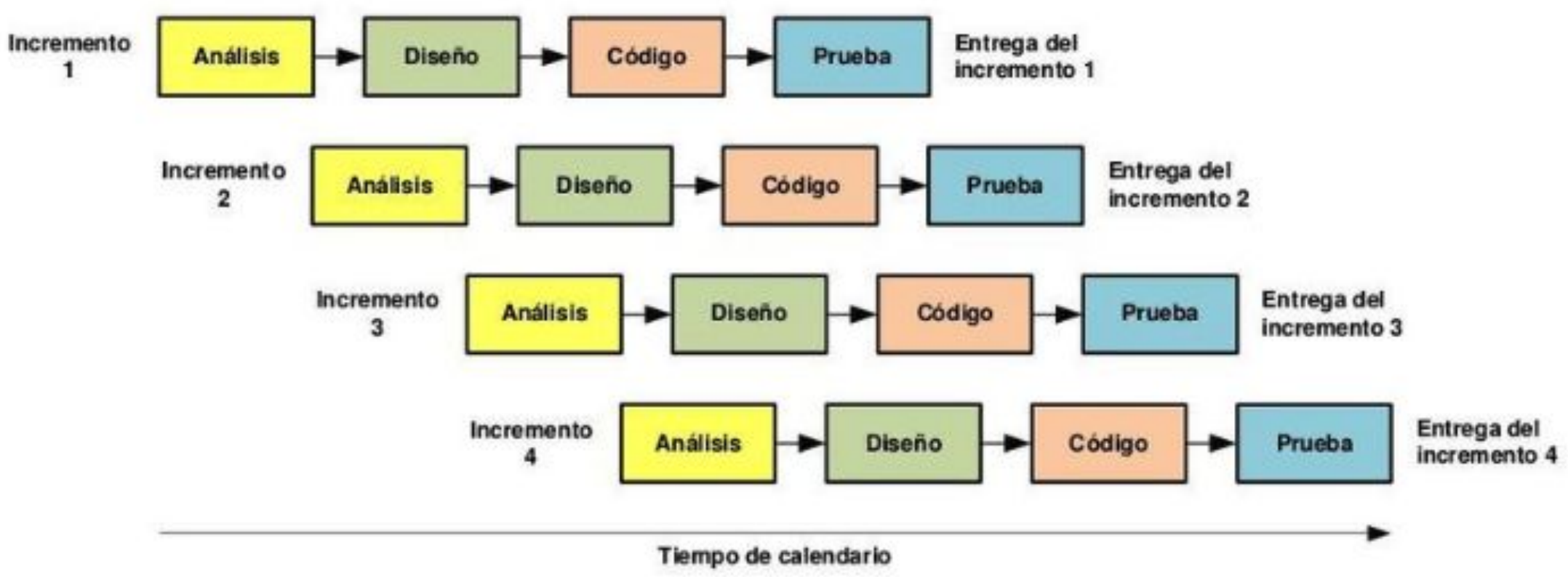
Toma las ventajas del modelo de desarrollo en cascada y el de prototipos añadiendo el concepto de análisis de riesgo.

- Se definen cuatro actividades:
  - **Planificación**: en la que se recolectan los requisitos iniciales o nuevos requisitos a añadir en esta iteración.
  - **Análisis de riesgo**: basándonos en los requisitos decidimos si somos capaces o no de desarrollar el software y se toma la decisión de continuar o no continuar.
  - **Implementación**: en el que se desarrolla un prototipo basado en los requisitos obtenidos en la fase de planificación.
  - **Evaluación del cliente**: el cliente comenta el prototipo. Si está conforme con él se acaba el proceso, si no se añaden los nuevos requisitos en la siguiente iteración.

# ESPIRAL

- Conjuga naturaleza iterativa construcción de prototipos con aspectos controlados y sistemáticos del modelo lineal secuencial.
- Proporciona potencial para el desarrollo rápido de versiones incrementales de SW.,
- Durante las primeras iteraciones, la versión incremental es un modelo papel o un prototipo.
- Durante las últimas iteraciones se producen versiones cada vez más completas del sistema desarrollado.
- Apto para :
  - Proy Mant Prod / Proy mejora Prod / Proy Des nuevos productos. / Proy Des de conceptos.

# INCREMENTAL



# INCREMENTAL

Permite construir el proyecto en etapas incrementales en donde cada etapa agrega funcionalidad.

Estas etapas, consisten en requerimientos, diseño, codificación, pruebas y entrega. Permite entregar al cliente un producto más rápido en comparación del modelo en cascada.

- Reduce los riegos ya que provee visibilidad sobre el progreso de las nuevas versiones.
- Provee retroalimentación a través de la funcionalidad mostrada.
- Permite atacar los mayores riesgos desde el inicio.
- Se pueden hacer implementaciones parciales si se cuenta con la suficiente funcionalidad.
- Las pruebas y la integración son constantes.
- El progreso se puede medir en periodo cortos de tiempo.
- Resulta más sencillo acomodar cambios al acortar el tamaño de los incrementos.
- Se puede planear en base a la funcionalidad que se quiere entregar primero.
- Por su versatilidad requiere de una planeación cuidadosa tanto a nivel administrativo como técnico.

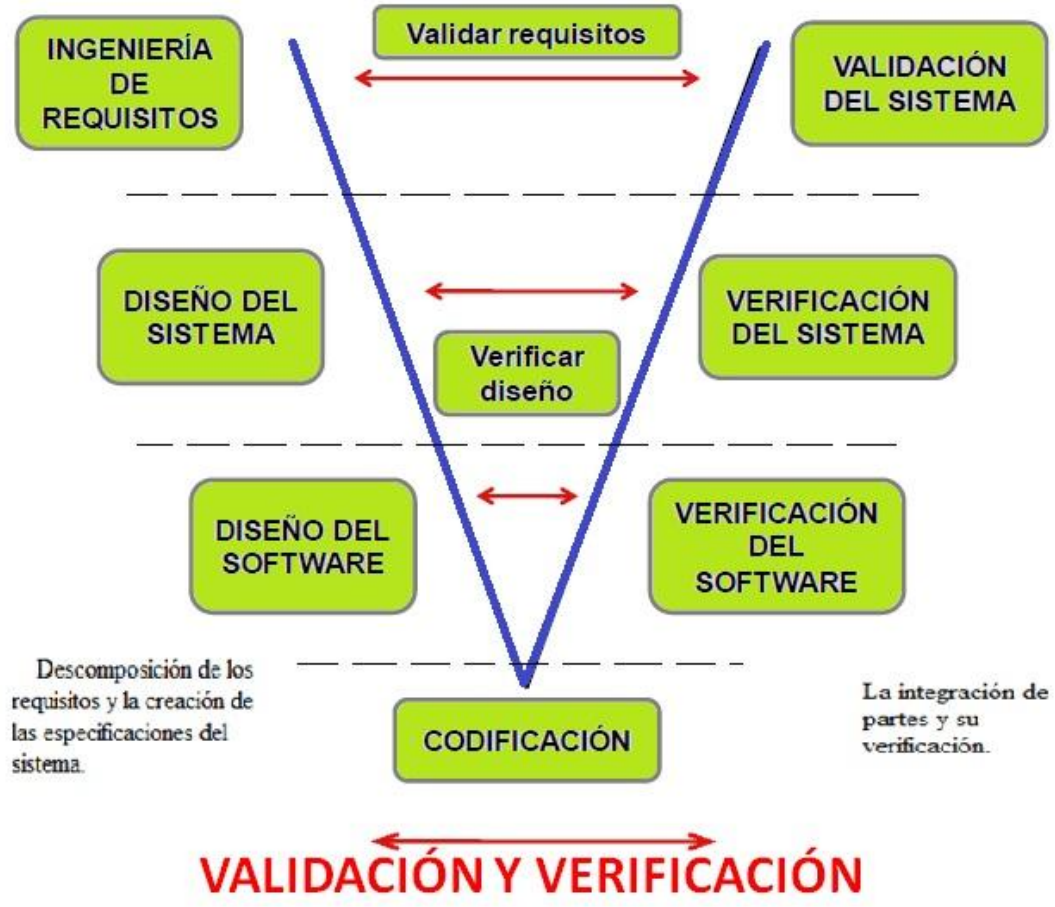
# INCREMENTAL

- **Ventajas**:
  - La solución se va mejorando en forma progresiva a través de las múltiples iteraciones, incrementa el entendimiento del problema y de la solución por medio de los refinamientos sucesivos.
  - Los clientes no esperan hasta el fin del desarrollo para utilizar el sistema. Pueden empezar a usarlo desde el primer incremento.
  - Los clientes pueden aclarar los requisitos que no tengan claros, conforme ven las entregas del sistema.
  - Se disminuye el riesgo de fracaso de todo el proyecto, ya que se puede distribuir en cada incremento.
  - Las partes más importantes del sistema son entregadas primero, por lo cual se realizan más pruebas en estos módulos y se disminuye el riesgo de fallos.

# INCREMENTAL

- **Desventajas**:
  - Requiere de mucha planeación, tanto administrativa como técnica
  - Requiere de metas claras para conocer el estado del proyecto.
  - Es un proceso de desarrollo de software, creado en respuesta a las debilidades del modelo tradicional de cascada.

# MODELO "V"


INGENIERÍA DE REQUISITOS
Validar requisitos
VALIDACIÓN DEL SISTEMA
DISEÑO DEL SISTEMA
Verificar diseño
VERIFICACIÓN DEL SISTEMA
DISEÑO DEL SOFTWARE
VERIFICACIÓN DEL SOFTWARE
Descomposición de los requisitos y la creación de las especificaciones del sistema.
CODIFICACIÓN
La integración de partes y su verificación.
VALIDACIÓN Y VERIFICACIÓN

# MODELO “V”

- Este modelo es un enfoque lineal secuencial que describe el flujo de trabajo desde la planificación hasta la implementación y la verificación del software. El modelo en V se llama así debido a su forma de "V", donde la parte superior del V representa la fase de planificación y la parte inferior del V representa la fase de verificación.
- El modelo en V se utiliza para asegurarse de que se cumplan los requisitos del cliente y que el software se entregue con la calidad esperada. En este modelo, las actividades de planificación, diseño y especificación se realizan en la parte superior del V, mientras que la implementación, pruebas y validación se realizan en la parte inferior. A medida que el proceso se mueve hacia abajo por el lado derecho del V, las pruebas y verificaciones van en aumento, lo que ayuda a identificar y corregir problemas antes de la entrega final del software.

# MODELO “V”

- Ventajas:
  - Enfoque estructurado: El Modelo en V proporciona un enfoque estructurado y fácil de entender para el desarrollo de software. El flujo de trabajo se divide en fases claras y bien definidas, lo que ayuda a los equipos de desarrollo a comprender claramente qué se espera de ellos en cada fase del proceso.
  - Verificación temprana de requisitos: El modelo en V ayuda a asegurar que los requisitos del cliente se entiendan y verifiquen temprano en el proceso de desarrollo. Esto puede ayudar a evitar errores costosos que podrían surgir más adelante en el proceso de desarrollo.
  - Enfoque de pruebas rigurosas: El modelo en V pone un fuerte énfasis en las pruebas y la validación del software. La parte inferior del V representa una serie de pruebas y validaciones que se realizan para asegurar que el software cumpla con los requisitos y funcione correctamente.

# MODELO “V”

- Desventajas:
  - Limitaciones de adaptabilidad: El Modelo en V puede ser inflexible y no permitir cambios de requisitos o funcionalidades a medida que avanza el proceso de desarrollo. Esto puede ser un problema en proyectos en los que los requisitos cambian con frecuencia.
  - Enfoque lineal: El modelo en V sigue un enfoque lineal secuencial, lo que significa que cada fase del proceso se completa antes de avanzar a la siguiente. Esto puede llevar a retrasos en el desarrollo y puede ser problemático si se descubren errores en fases posteriores del proceso.
  - Falta de colaboración: El modelo en V se enfoca en roles y responsabilidades específicos para cada fase del proceso de desarrollo, lo que puede limitar la colaboración entre los miembros del equipo. Esto puede ser un problema en proyectos donde se requiere una estrecha colaboración entre los equipos de desarrollo y los interesados.

# Proyecto Industria Logistica

Keep it simple, stupid (KISS) is **a design principle which states that designs and/or systems should be as simple as possible**. Wherever possible, complexity should be avoided in a system—as simplicity guarantees the greatest levels of user acceptance and interaction.

- Migración a Cloud de los Sistemas Actuales
- Front End para el eCommerce
- Back End para el eCommerce
- Implementación del eCommerce en Cloud
- Integración del eCommerce con el Sistema Legacy
- Migración hacia un nuevo ERP (Enterprise Resource Planning)
- QA para el eCommerce
- Arquitectura del eCommerce
- Costos y Presupuestos de Herramientas para el eCommerce
- Gestión del Proyecto con Entregables
- B2B con WalMart
- Seguridad del eCommerce
- Diseño Gráfico del eCommerce
- Gestión de Datos del eCommerce

# GRACIAS!