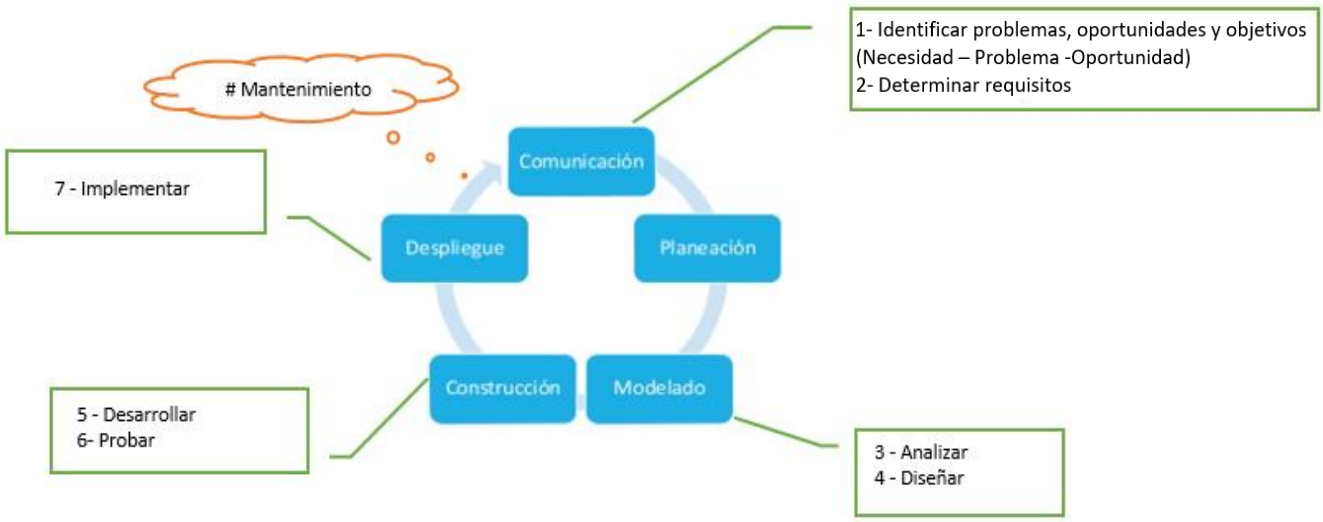
Ciclos de Vida del Software

Actividades estructurales del proceso (SDLC: Ciclo de Vida del Desarrollo de SW)

Tienen un conjunto de actividades que varían de acuerdo con el autor que las desarrolle. Conceptualmente todas describen el mismo proceso, aunque con distinta estructura de agrupamiento.

Se segmenta en 3, 5 u 8 actividades o fases de acuerdo con el autor. Nosotros nos basamos en lo definido por Pressman.



Comunicación. Antes de que comience cualquier trabajo técnico, tiene importancia crítica comunicarse y colaborar con el cliente (y con otros participantes). Se busca entender los objetivos de los participantes respecto del proyecto, y reunir los requerimientos que ayuden a definir las características y funciones del software.

Planeación. Cualquier viaje complicado se simplifica si existe un mapa. Un proyecto de software es un viaje difícil, y la actividad de planeación crea un "mapa" que guía al equipo mientras viaja. El mapa —llamado *plan del proyecto de software*— define el trabajo de ingeniería de software al describir las tareas técnicas por realizar, los riesgos probables, los recursos que se requieren, los productos del trabajo que se obtendrán y una programación de las actividades.

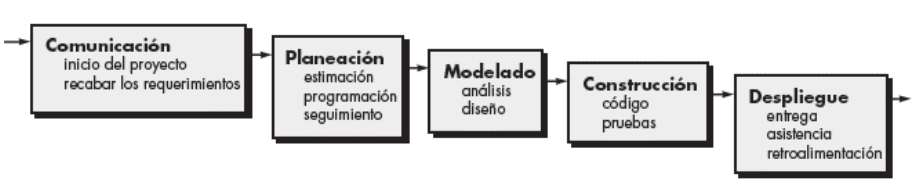
Modelado. Ya sea usted diseñador de paisaje, constructor de puentes, ingeniero aeronáutico, carpintero o arquitecto, a diario trabaja con modelos. Crea un "bosquejo o boceto" del objeto por hacer a fin de entender el panorama general —cómo se verá arquitectónicamente, cómo ajustan entre sí las partes constituyentes y muchas características más—. Si se requiere, refina el bosquejo con más y más detalles en un esfuerzo por comprender mejor el problema y cómo resolverlo. Un ingeniero de software hace lo mismo al crear modelos a fin de entender mejor los requerimientos del software y el diseño que los satisfará. Análisis – Diseño – Implementación –Pruebas.

Construcción. Esta actividad combina la generación de código (ya sea manual o automatizada) y las pruebas que se requieren para descubrir errores en éste.

Despliegue. Se orienta a la implantar el sistema en el dominio definido. Incluye las acciones de entrega, capacitación, retroalimentación y mantenimiento.

1) Modelos Prescriptivos

1.1) CICLO DE VIDA EN CASCADA (Royce 1970)



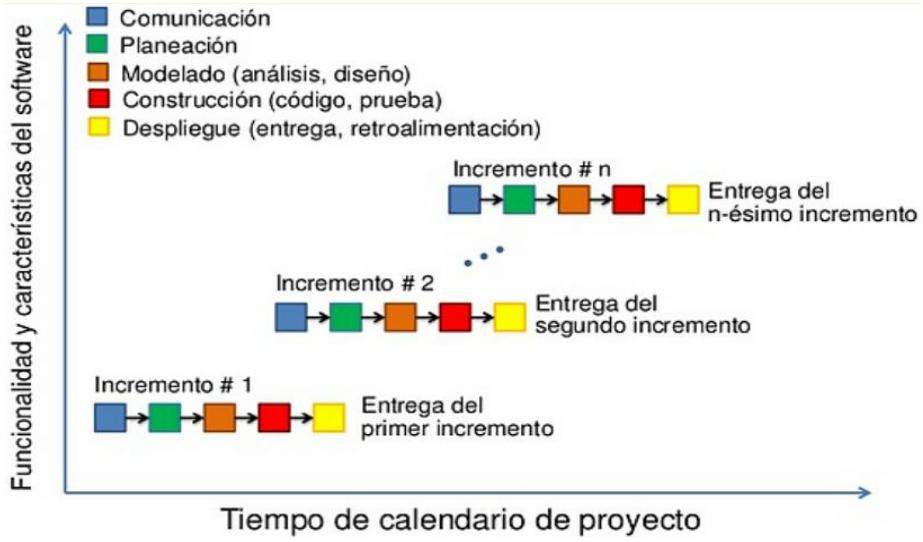
Después de cada etapa realiza una o varias revisiones para comprobar si se puede pasar a la siguiente. Es un modelo rígido, poco flexible y con muchas restricciones. Aunque fue uno de los primeros sirvió de base para el resto de los ciclos de vida. Buena opción cuando la modificación de los requerimientos tiene nula probabilidad de ocurrencia. Los requerimientos deben estar perfectamente definidos. Las etapas son consecutivas y no simultaneas.

Ventajas	Desventajas
Define claramente las distintas etapas y su transición.	Se congelan los requisitos al principio
Permite al PM controlar el avance	Se entrega el producto al final
Es corto y económico	Desaprovechamiento de recursos
Permite validar las actividades dentro de una misma etapa	Si no cumple las expectativas hay que iniciar de cero
	Imposible retroceder

1.2) CICLO DE VIDA INCREMENTAL (Hisch 1985)

Deriva del ciclo de vida en cascada, busca reducir el riesgo que surge entre las necesidades del usuario y el producto final por malentendidos durante la etapa de requerimientos. El cliente puede ir mejorando los requerimientos definidos, pero no puede agregar.

Divide el proyecto en miniproyectos con funcionalidades enteras.



Es la iteración de varias cascadas, al final de cada iteración se entrega una versión con mayor funcionalidad del producto.

Las necesidades se plantean al principio, pero no se congelan.

No se agregan nuevas funcionalidades, pero si puede modificarse algo que aún no se hizo, estos cambios se dejan para el final.

Refuerzan buenos hábitos, conocer antes de definir, definir antes de diseñar, diseñar antes de codificar.

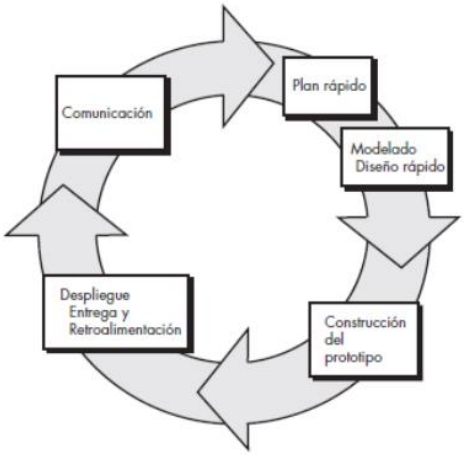
Los clientes no tienen que esperar hasta que se entregue todo el sistema antes de que puedan obtener valor de él.

Ventajas	Desventajas
Aprovechamiento de los RRHH	Poca adaptación a los cambios.
Fácil entendimiento e Implementación	

	No se pueden agregar funcionalidades ni cambiar las que ya están terminadas.
Si el producto básico es bien recibido, entonces se agrega más personal (si se requiere) para que participe en el siguiente incremento	Una primera entrega apresurada puede no satisfacer el producto fundamental y condicionar el resto del desarrollo - Riesgo Moderado de soslayar la arquitectura (puede ser endeble y colapsar en iteraciones posteriores).

1.3) CICLO DE VIDA PROTOTIPADO (EVOLUTIVO PROTOTIPO)

Los modelos evolutivos son iterativos. Se caracterizan por la manera en la que permiten desarrollar versiones cada vez más completas del software.



Se utiliza cuando el cliente no sabe bien lo que quiere, se arma una maqueta o prototipo, puede ser de 3 tipos:

Maqueta: en papel o pantalla hueca. Se hace solo para mejorar la definición, después se elige un ciclo de vida. (NO ES UN CV)

Desechable: tiene algo de funcionalidad para que el cliente vea luego se desecha y se elige un ciclo de vida. (NO ES UN CV)

Evolutivo: es un ciclo de vida, comienza con el prototipo, con funcionalidad, lo muestro al cliente y de acuerdo con las modificaciones va evolucionando, se hace sobre lo que más se entiende, evoluciona hasta completar el sistema.

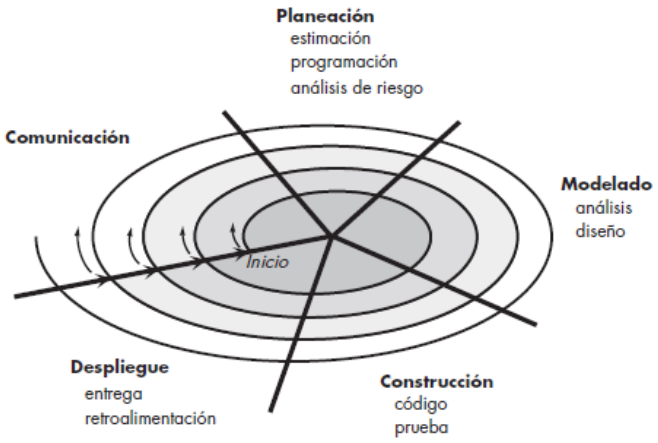
En cada Ciclo tengo obligatoriamente una funcionalidad completa

Progresivamente se incrementa la definición de funcionalidades manteniéndose el alcance en líneas generales constante. Es muy útil cuando hay que liberar entregas tempranas.

Ventajas	Desventajas
Permite agregar funcionalidades y cambiar algunas de las ya hechas (con una buena gestión de cambios)	Es largo y costoso
El cliente ve prontamente una parte del sistema.	Hay que controlar los cambios para que el sistema no se haga eterno.
Permite que el grupo de desarrollo entienda mejor la funcionalidad y que el cliente exprese mejor sus necesidades	Grave Riesgo de que se soslaye la arquitectura (puede ser endeble y colapsar en iteraciones posteriores).
Aprovecha los RRHH (cuando termina el análisis de una parte del sistema se empieza a analizar otra)	

1.4) CICLO DE VIDA MODELO EVOLUTIVO EN ESPIRAL (Bohem 1988)

Modelo espiral es un modelo evolutivo del proceso del software y se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistémicos del modelo de cascada. Tiene el potencial para hacer un desarrollo rápido de versiones cada vez más completas, *siendo de importancia la gestión del riesgo*.



Con el empleo del modelo espiral, el software se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, lo que se entrega puede ser un modelo o prototipo. En las iteraciones posteriores se producen versiones cada vez más completas del sistema cuya ingeniería se está haciendo.

Cada vuelta es un planteo de necesidades y una parte del sistema. Luego se hace la evaluación con el cliente y se evalúa si satisface las primeras necesidades, sino se hace otra vuelta con las mismas necesidades.

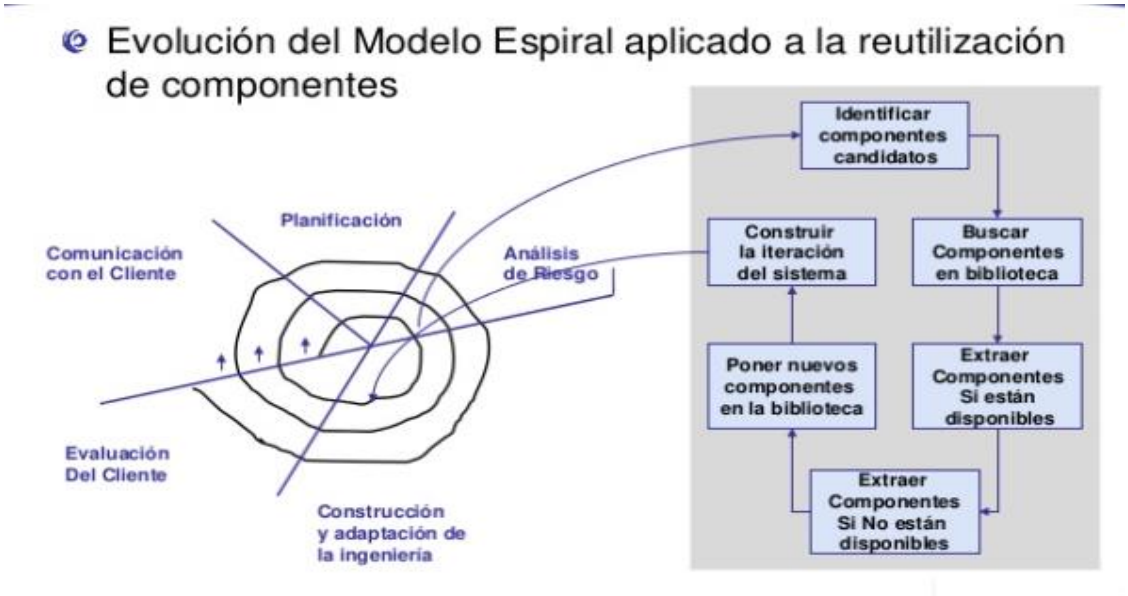
Este modelo combina más de un ciclo de vida, una cascada dentro de la construcción del prototipo, también puede haber un incremental o prototipo evolutivo dependiendo si se hace parte de la funcionalidad o la función completa y también dependiendo de lo que hago en la primer vuelta, si agrego nuevas funcionalidades o no. En la mayoría de los casos hay escondido un ciclo evolutivo.

Ventajas	**Desventajas**
Tiene en cuenta los **riesgos** durante todo el desarrollo del proyecto	Es largo y costoso más que el evolutivo.

2) MODELO DE ENSAMBLAJE DE COMPONENTES (Nie 1992)

Se ve que componentes se necesita en cada vuelta, se buscan estos en la biblioteca si están se usan y si es necesario se crean.

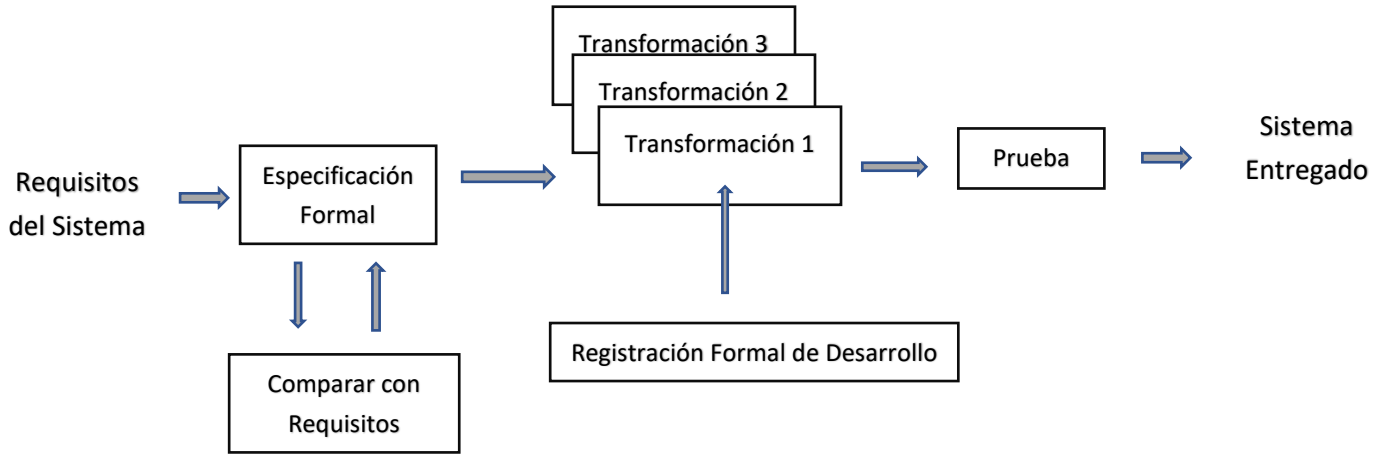
Ventajas	Desventajas
Optimiza tiempos y costos Facilita el mantenimiento	No se dispongan los componentes adecuados para los requisitos del sistema. Si las nuevas versiones de los componentes no están bajo control de quien los utiliza se pierde parte de la evolución del sistema.



3) MODELO DE TRANSFORMACION FORMAL (Dijkstra 1976)

Se basa en la transformación de una especificación formal a lo largo de varias representaciones hasta llegar a un programa ejecutable/app.

Las transformaciones preservan la corrección, permiten comprobar fácilmente que el programa es conforme a la especificación



Evolución del Desarrollo por Iteración en Modelos Prescriptivos

	Incremental			Evolutivo Prototipo			Evolutivo Espiral			Cascada		
Funcionalidad Versión	A	B	C	A	B	C	A	B	C	A	B	C
1	100%	0%	0%	30%	30%	30%	60%	30%	0%	0%	0%	0%
2	100%	100%	0%	60%	60%	60%	100%	66%	50%	0%	0%	0%
3	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Describo 1 Funcionalidad por Vez | **Voy Incrementando el Conocimiento por Ciclo** | **Describo Todo de una Vez**

Ingeniería del Software - Capas de la Ingeniería de Software, (2005)

"El fundamento de la ingeniería del software es la capa de proceso" (Pressman, 2005)

"Las nociones fundamentales de procesos y la organización del sistema son la base de todas estas técnicas y éstas son la esencia de la ingeniería del software". (Sommerville, 2005)

Lo principal es el proceso (proporciona la base para el control de la administración de proyectos SW, establece el contexto en el que se aplican métodos técnicos, se generan productos se establecen puntos de referencia, se asegura la calidad y se administra el cambio de manera apropiada.

Un proceso define quien hace, qué, cuándo y como, para alcanzar un objetivo (Jacobson, Booch y Rumbaugh)