1. Introducción
2. Sistema informático
3. Lenguajes de programación
4. Formas de ejecución de los programas: tipos de lenguajes
5. Desarrollo de software

¿Qué es la informática?

- Según la RAE
  - Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de computadoras

¿Qué es un ordenador?

- "Un sistema digital con tecnología microelectrónica capaz de procesar información a partir de un grupo de instrucciones denominado programa"
- Componentes principales:
  - Procesador
  - CPU: Unidad central de procesos
  - ALU: Unidad aritmética y lógica
  - Memoria (RAM: Random Access Memory)
- Componentes auxiliares:
  - Disco duro
  - Lector CD-ROM
  - Teclado
  - Pantalla

Representación de los datos en el ordenador

- En un principio todos los programas eran creados por el único código que era capaz el ordenador de entender: **el código máquina.**
- El código máquina es un conjunto de 1s y 0s de grandes proporciones.
- Este método de programación convertía la labor de programación en una labor tediosa.

Representación de textos

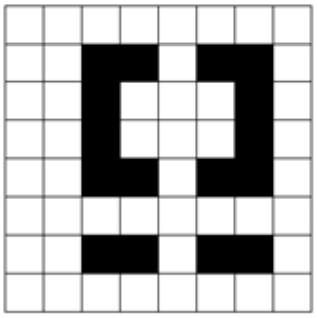
- Texto: Me llamo Iván
- ASCII: 077 101 032 108 108 097 109 111 032 073 118 097 110
- 13 bytes

Caracteres - La tabla ASCII

32		33	!	34	"	35	#	36	$	37	%
38	&	39	'	40	(	41	)	42	*	43	+
44	,	45	-	46	.	47	/	48	0	49	1
50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=
62	>	63	?	64	@	65	A	66	B	67	C
68	D	69	E	70	F	71	G	72	H	73	I
74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U
86	V	87	W	88	X	89	Y	90	Z	91	[
92	\	93	]	94	^	95	_	96	`	97	a
98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m
110	n	111	o	112	p	113	q	114	r	115	s
116	t	117	u	118	v	119	w	120	x	121	y
122	z	123	{	124	—	125	}	126	~		

Representación de imagen

- División de la imagen en una matriz de pixeles (unidad de la imagen)
- Cada pixel asociado con un color



```
00000000
00110110
00100010
00100010
00110110
00000000
00110110
00000000
```

¿Qué es un programa?

- Un programa es una secuencia de instrucciones que ejecuta la CPU de manera secuencial con el objetivo de realizar varias tareas.
- Se almacena junto a los datos y resultados intermedios y definitivos en la memoria (RAM).
- Cada instrucción es un conjunto de bytes
- Ejemplos de instrucciones:
  - Leer un dato del teclado
  - Guardar un dato en la memoria
  - Ejecutar una operación sobre dos datos
  - Mostrar un dato en la pantalla

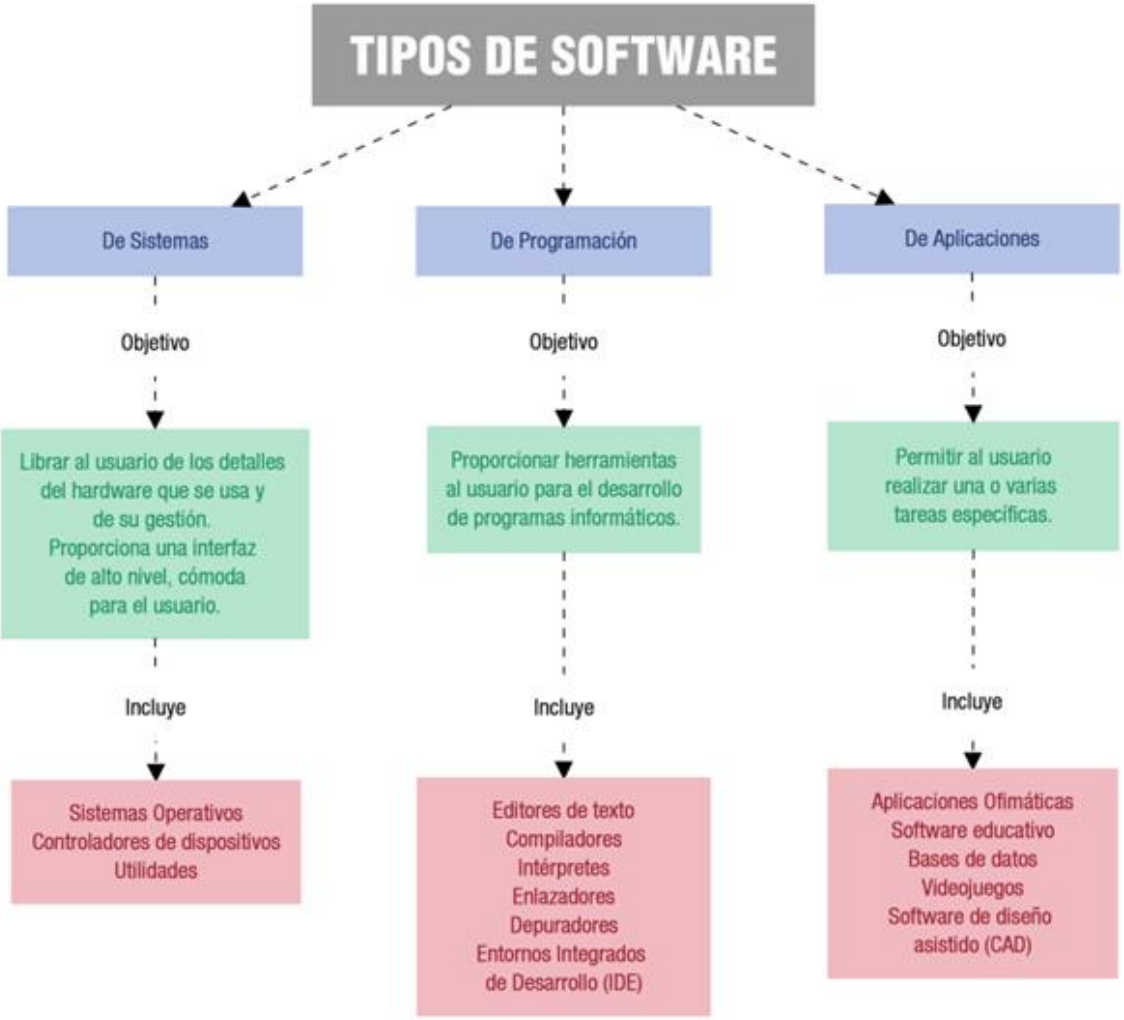
Sistema informático

- Conjunto de cosas que ordenadamente relacionadas entre sí contribuyen a un fin.
- Compuestos por ordenadores y sus periféricos. Partes:
  - **Hardware**, son los elementos materiales, los que se pueden tocar.
  - **Software**, los programas que gobiernan el funcionamiento del computador.
- Objetivo:
  - Tratamiento de la información: almacenamiento, elaboración y presentación de datos.

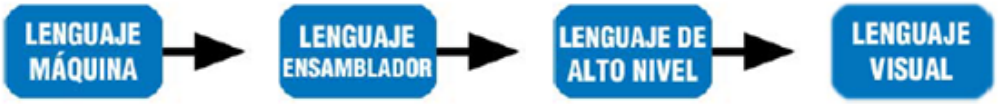
Conjunto de programas del ordenador.

Se distingue 3 tipos de software:

- Sistema operativo
- Software de programación
- Aplicaciones


TIPOS DE SOFTWARE
De Sistemas
De Programación
De Aplicaciones
Objetivo
Objetivo
Objetivo
Librar al usuario de los detalles del hardware que se usa y de su gestión. Proporciona una interfaz de alto nivel, cómoda para el usuario.
Proporcionar herramientas al usuario para el desarrollo de programas informáticos.
Permitir al usuario realizar una o varias tareas específicas.
Incluye
Incluye
Incluye
Sistemas Operativos
Controladores de dispositivos
Utilidades
Editores de texto
Compiladores
Intérpretes
Enlazadores
Depuradores
Entornos Integrados de Desarrollo (IDE)
Aplicaciones Ofimáticas
Software educativo
Bases de datos
Videojuegos
Software de diseño asistido (CAD)

- Conjunto de instrucciones, operadores y reglas de sintaxis y semánticas para llevar a cabo tareas en el ordenador.
- El idioma artificial que constituye los operadores, instrucciones y reglas tienen el objetivo de facilitar la tarea de crear programas.
- Los programas controlan el comportamiento del ordenador
- Los lenguajes de programación han sufrido su propia **evolución:**



- **1ª generación**: muy próximos al lenguaje de la máquina.
  - Lenguajes máquina → secuencias de 1 y 0
  - Lenguajes ensamblador → asocia a cada secuencia de 1s y 0s de la máquina una instrucción para realizar operaciones sencillas.
- **2ª generación**: lenguajes de programación de propósito general con un alto nivel de abstracción. Programación mas entendible e intuitiva.
  - Fortran → aplicaciones científicas
  - Cobol → aplicaciones de gestión empresarial.
  - Basic → PCs, sencillo y fácil de aprender.

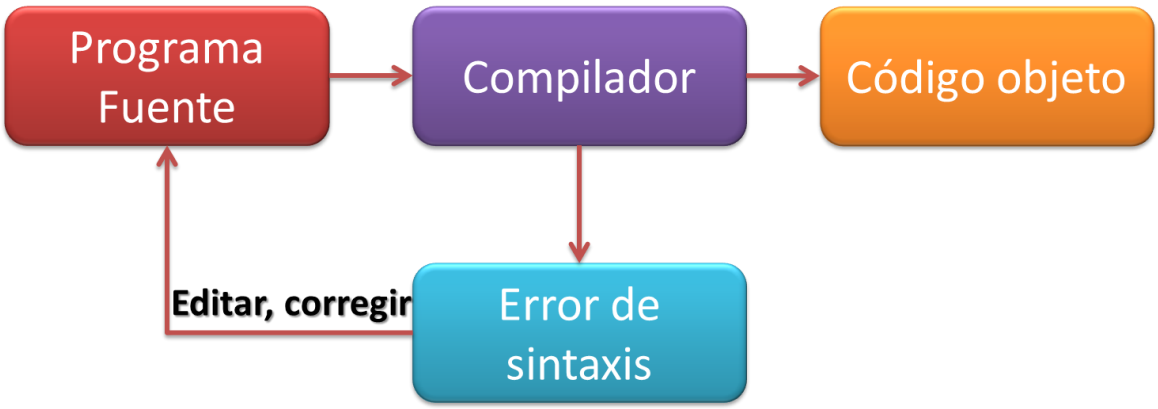
- **3ª generación:** Surge la programación estructurada que facilita la realización de programas fáciles de leer, escribir y entender.
  - PASCAL, fue diseñado para la **enseñanza** de la programación estructurada.
  - MÓDULA-2, descendiente de pascal, se incorpora la estructura de **módulo**. Se mejora modularidad, concurrencia, abstracción y ocultación
  - C, desarrollado para la codificación del UNIX. Flexible y potente. No hay restricciones sobre las operaciones con distintos tipos.
  - ADA, descendiente de pascal, mucho más potente y complejo. Incorpora modularidad, abstracción, ocultación, concurrencia y sincronización.
  - SMALLTALK, precursor de los lenguajes orientados a objetos (LOO). La POO es una forma especial de programación que expresa las cosas mas cercana a nuestra forma de pensar.
  - C++, incorpora en C los mecanismos de la POO.
  - JAVA, incorpora mecanismos de la POO, parecido a C pero mas simple y multiplataforma.
  - LISP, lenguaje **funcional** usado en IA y sistemas expertos.
  - PROLOG, lenguaje lógico en que se construye una base de conocimiento basada en reglas a partir de la cual podemos inferir nuevos hechos o reglas.

- **4ª generación**: mayor grado de abstracción. Se alejan aún mas de la máquina. Se centran en la resolución del problema.
  - BASES DE DATOS; como SQL permiten acceder y manipular la información.
  - GENERADORES DE PROGRAMAS, son eficientes en un dominio de aplicaciones limitado. La mayoría producen aplicaciones de gestión.

- Dependiendo de cómo un programa se ejecute dentro de un sistema, tenemos 3 categorías de lenguajes:
  - Lenguajes compilados
  - Lenguajes interpretados
  - Lenguajes virtuales

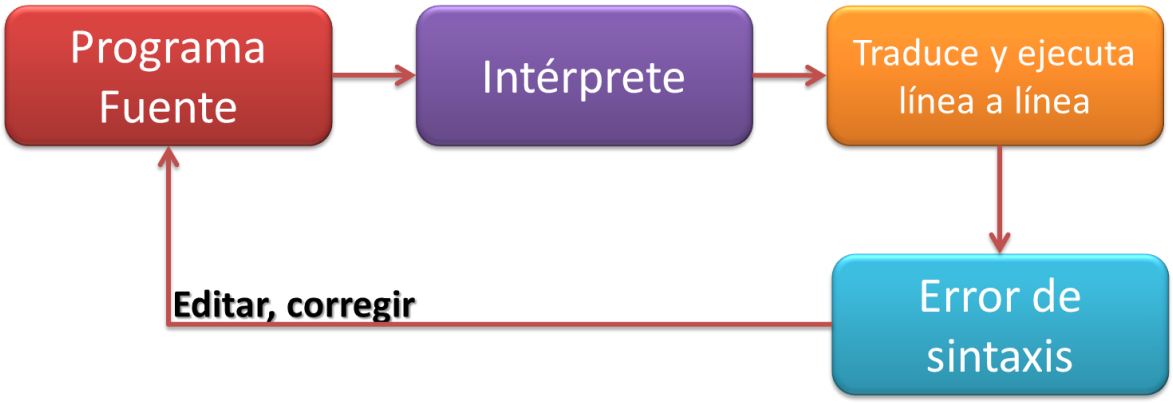
Lenguajes compilados

- El lenguaje compilado realiza el proceso de conversión de fuente a objeto.
- Un programa enlazador unirá el código objeto con las librerías necesarias para producir el código ejecutable.
- Luego se realizará la ejecución del mismo.
- Ejemplos: Fortran, Familia de lenguaje C, incluyendo C++, Objective C, Ada, Pascal, Algol.



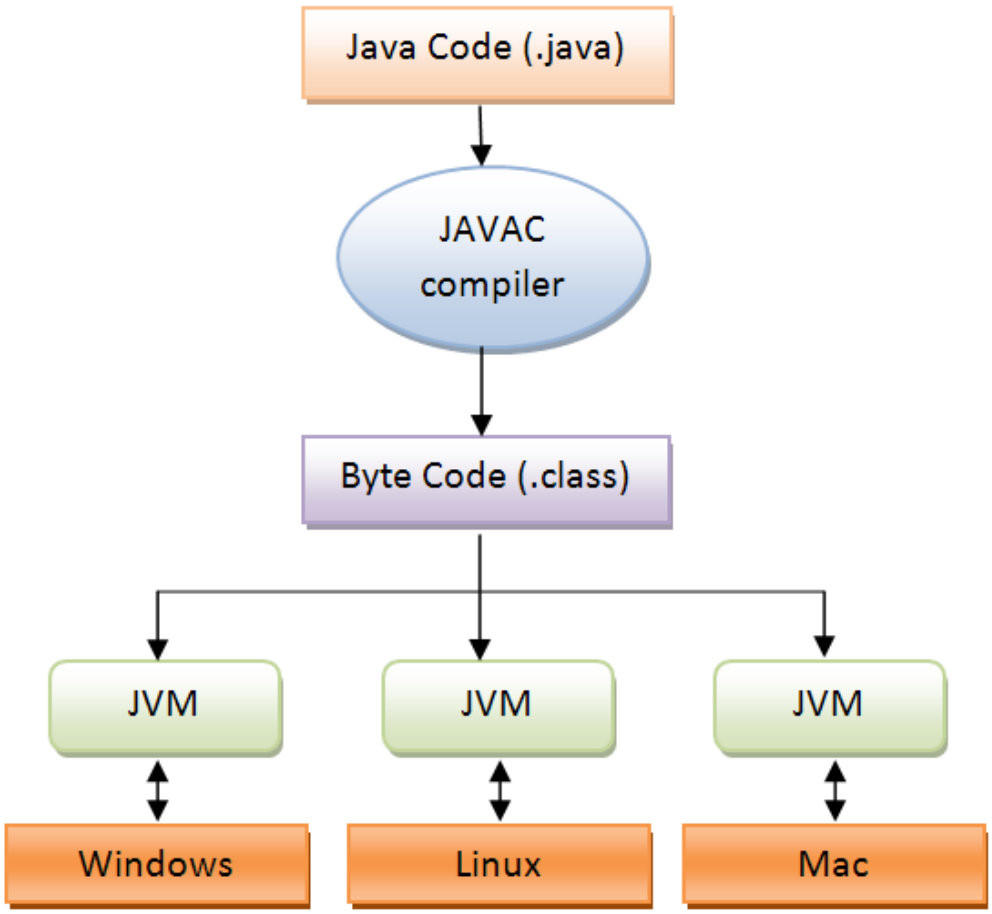
Lenguajes interpretados

- En un lenguaje interpretado, las instrucciones son traducidas y ejecutadas línea a línea.
- Ejemplos: Perl, PHP, Cobol, ActionScript, ASP, Bash, etc.


Programa Fuente
Intérprete
Traduce y ejecuta línea a línea
Error de sintaxis
Editar, corregir

Lenguajes virtuales

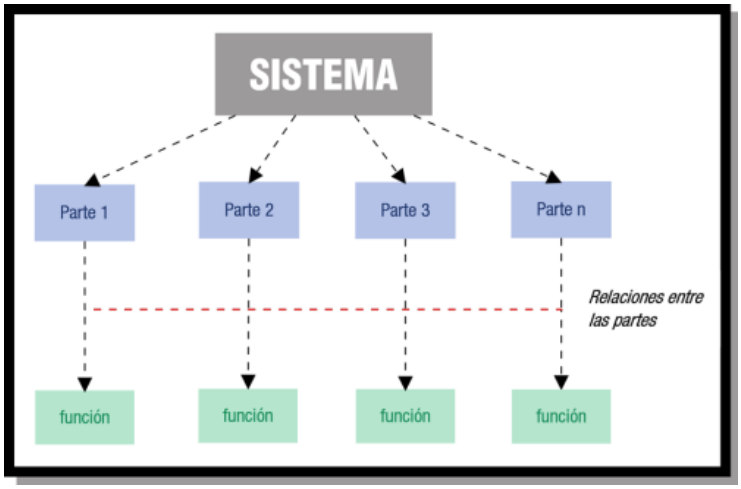
- Funcionamiento similar a lenguajes compilados pero en vez de generar un código objeto genera un "bytecode".
- El bytecode puede ser interpretado por cualquier arquitectura que tenga la **máquina virtual** correspondiente.



- Proceso desde que se concibe una idea hasta que un programa está en el ordenador.
- **No** se plantea solo una actividad de programación
- Consta de una serie de pasos de obligado cumplimiento: **CICLOS DE VIDA DEL SOFTWARE**

- Es la **primera** fase del proyecto. Posiblemente sea la fase **más complicada**. Depende de pericia del **analista**.
- Se determina **qué** debe hacer el software. Debe existir una buena **comunicación** entre en analista y el cliente
- Se especifican los requisitos funcionales y no funcionales del sistema.
  - **Funcionales**: qué funciones tendrá que realizar la aplicación. Qué respuesta dará la aplicación ante todas las entradas. Cómo se comportará la aplicación en situaciones inesperadas.
  - **No funcionales**: tiempos de respuesta del programa, legislación aplicable, tratamiento ante la simultaneidad de peticiones, etc.
- Documento: **ERS** (Especificación de requisitos del software): especificación precisa y completa a partir de los requisitos establecidos por el cliente.

- Durante esta fase, donde ya sabemos lo que hay que hacer, el siguiente paso es ¿**cómo** hacerlo?
- Se divide el sistema en partes y se establece relación hay entre ellas
- Decidir qué hará exactamente cada parte.
- Decisiones: entidades y relaciones de la base de datos, lenguaje de programación, selección del SGBD...
- Documento: Documento de Diseño del Software (SDD),descripción de la estructura global del sistema, especificación de qué debe hacer cada uno de los módulos y de cómo se combinan



- Se elige un lenguaje de programación.
- Se codifican los programas, es decir, se realiza el proceso de programación.
- Es tarea del programador y **tiene que cumplir todo lo impuesto en las fases de análisis y diseño** de la aplicación.
- Durante esta fase, el código pasa por los 3 estados visto antes:
  - Código fuente
  - Código objeto*
  - Código ejecutable

- **Recuerda que en los lenguajes interpretados no se produce código objeto. El paso de fuente a ejecutable es directo*

- Una vez obtenido el software, la siguiente fase del ciclo de vida son las pruebas (aunque existen metodologías en las que el diseño de las pruebas es previo a la codificación)
- Se prueban los programas para detectar errores y se depuran.
- Se realizan sobre un conjunto de datos de prueba.
- **Imprescindible** para asegurar la **verificación y validación** del software construido.
- Incluyen pruebas unitarias y pruebas de integración.
  - **Unitarias**: prueban **una por una** las distintas partes de software y comprueba su funcionamiento
  - **De integración**: una vez que se realizan con éxito las unitarias se debe probar el funcionamiento del sistema completo
- La última prueba se denomina Beta Test y se realiza en entornos de producción.

- De **todas las etapas se documentan** y guarda toda la información. Por tanto es una etapa **transversal** al resto.
- Documentación a elaborar en el proceso de desarrollo:
  - Guía técnica (Análisis, diseño, codificación y pruebas)
  - Guía de uso (Descripción aplicación, forma de ejecutar la aplicación, ejemplos de uso, requerimientos de software, solución de posibles problemas que se puedan presentar,..).
  - Guía de instalación (Puesta en marcha, explotación y seguridad).

- Después de las fases anteriores, una vez realizadas las pruebas y documentadas todas las fases, el siguiente paso es la **explotación**.
- **Instalamos, configuramos y probamos** la aplicación en los equipos del cliente.
- Los usuarios finales conocen la aplicación y comienzan a utilizarla.
- Se lleva a cabo la BETA TEST en los propios equipos cliente y bajo cargas normales.
- **Momento crítico del proyecto.** Importante tenerlo todo preparado antes de presentarle el producto al cliente

PREGUNTA: ¿Con la entrega del proyecto hemos terminado nuestro trabajo?

- La respuesta es NO

- Es la etapa mas larga de todo el ciclo de vida del software.
- Se define como el proceso de control, mejora y optimización del software.
- Se mantiene el contacto con el cliente para actualizar y modificar la aplicación en el futuro.
- Siempre surgen errores que habrá que ir corrigiendo y nuevas versiones del producto.
- Se pacta con el cliente un servicio de mantenimiento de la aplicación.

- Inicialmente la tarea de desarrollo realizada **individualmente** de forma **poco disciplinada**.
- Aparece cuando surge la necesidad de desarrollar aplicaciones software demasiado complejas (60s).
- Ejemplos:
  - Muertes por el Therac-25 (1985-1987)
  - Sobrecosto, retraso y cancelación en el sistema del Bank of America (1988)
  - Accidente del Ariane 5 (1996)
- Para superar la crisis:
  - Aparición de metodologías de desarrollo.
  - Concepción de la Ingeniería del Software como disciplina.
  - Trabajo en equipo: división, especialización del trabajo (analistas, programadores, ...)

- Históricamente se han utilizado técnicas como:
  - El modelado
  - División del Producto
  - División del Proceso
- En principio se deberían utilizar estas técnicas, también en informática.

El modelado de sistemas

- Representa el objeto a construir
- Ayuda a tratar la complejidad de los sistemas.
- Ayuda al ingeniero a visualizar el sistema a construir.
- Ayudan a verificar la corrección del sistema.

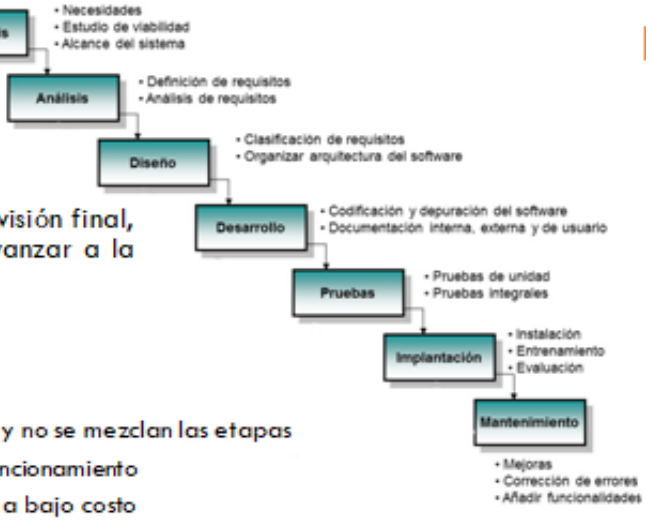
División del producto

- Se fracciona el producto de modo que cada fragmento lo puede realizar un miembro del grupo de desarrollo.

- **División del proceso**
  - Implica dividir el desarrollo del artefacto o aplicación por fases. Normalmente se habla de especificación, diseño y fabricación.

- **Ciclo de vida**
  - Conjunto de pasos a seguir para desarrollar un programa.
  - También se puede definir como el **proceso de desarrollo y mantenimiento del software**.
  - Consiste en determinar:
    - Las **fases** productivas de un proyecto.
    - Los **objetivos** de cada fase productiva.
    - Los **productos** obtenidos en cada una de estas fases así como sus **características**.
- **Siempre se debe aplicar un modelo de ciclo de vida** al desarrollo de cualquier proyecto software serio.
- Según el modelo elegido se describen un conjunto de actividades para llevar a cabo el ciclo de vida
- Los modelos **clásicos**:
  - Modelo en cascada
  - Modelo en cascada con retroalimentación
- Los modelos **evolutivos**:
  - Modelo incremental
  - Modelo en espiral
- Los modelos **ágiles**:
  - eXtreme Programming (XP)
  - Scrum
- Prácticamente identifican actividades similares y sólo se diferencian en la forma de presentación

Modelo en cascada

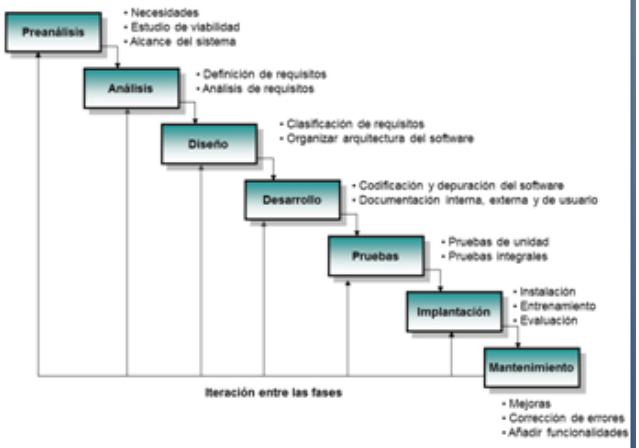


- Ordena las etapas del proceso de tal forma que el inicio de cada etapa debe **esperar a la finalización** de la etapa anterior.
- Al final de cada etapa, se lleva a cabo una revisión final, que determina si el proyecto está listo para avanzar a la siguiente fase.
- Fue el **primero** en originarse

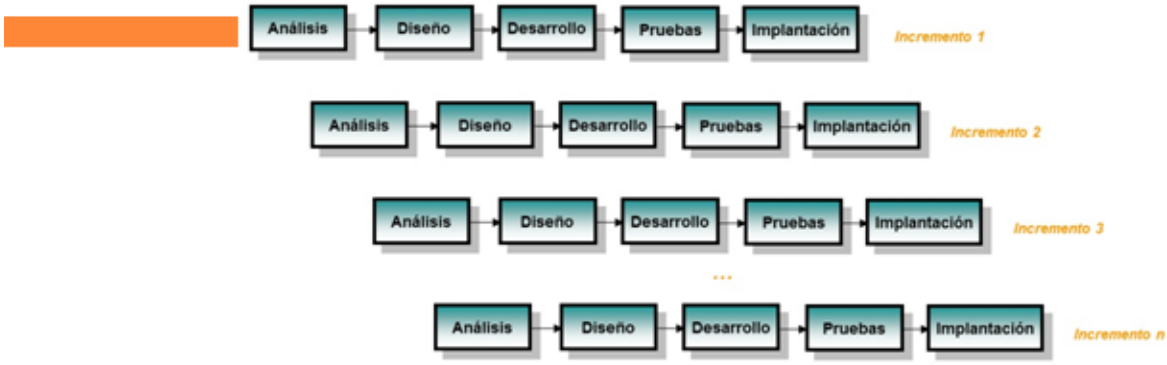
- **Ventajas**
  - Sencillo y disciplinado. Se tiene todo organizado y no se mezclan las etapas
  - Fácil de aprender a utilizarlo y comprender su funcionamiento
  - Ayuda a detectar errores en las primeras etapas a bajo costo
  - Ayuda a minimizar los gastos de planificación, pues se realiza sin problemas
- **Inconvenientes**
  - Los proyectos **raramente siguen el proceso lineal** tal como se definía originalmente el ciclo de vida
  - Es **difícil** que el cliente exponga explícitamente todos **los requisitos al principio**
  - El cliente debe tener **paciencia** pues obtendrá el producto al final del ciclo de vida. Proceso lento y pesado
  - Puede **resultar complicado** regresar a etapas anteriores (ya acabadas) para realizar **correcciones**
  - El producto final obtenido puede que no refleje todos los requisitos del usuario

Modelo en cascada con retroalimentación

- Similar al anterior pero se produce retroalimentación entre etapas.
- Tiene la **ventaja** que así se ofrece la posibilidad de realizar cambios o evoluciones
- Permite saltar de una etapa a la anterior, o incluso saltar a otras anteriores si es requerido
- **Inconvenientes**
  - El problema es que los cambios siguen siendo muy costosos
  - Es por ello que los clientes deben tener los requisitos muy claros desde el principio (cosa muy infrecuente)
  - Seguimos sin tener un producto final hasta que termine el ciclo de vida



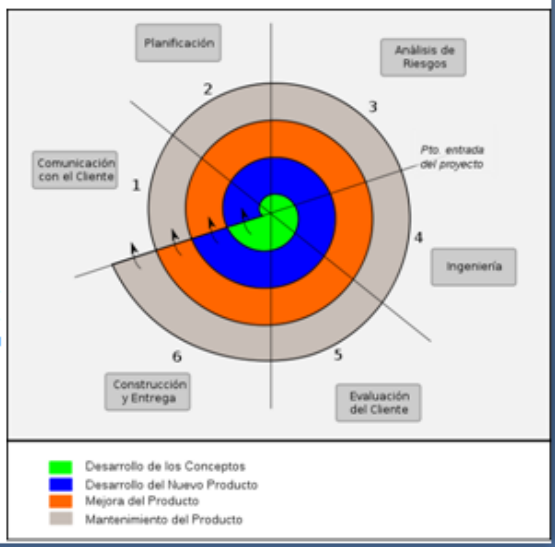
Modelo incremental



- Es un modelo de tipo evolutivo que está basado en varios ciclos de vida en cascada realimentados aplicados repetidamente, con una filosofía iterativa
- Este modelo emplea secuencias lineales escalonadas que proporcionan **incrementos** del producto en el que se añaden nuevas funcionalidades en cada uno de ellos
- **Ventajas:**
  - Se entrega algo de valor a los clientes cada cierto tiempo y se reduce el tiempo de desarrollo inicial
  - La evaluación del incremento por parte del cliente, original un plan para el siguiente incremento.
  - El cliente se involucra más
- **Inconvenientes:**
  - Requiere de mucha planificación, tanto administrativa como técnica
  - Difícil de evaluar el coste total. Además requiere de gestores experimentados

Modelo en espiral

- En cada giro se construye un modelo del sistema completo. Cada entrega es más evolucionada que la anterior
- Puede combinarse con otros modelos de proceso de desarrollo.
- Buen modelo para el desarrollo de **grandes sistemas**.
- No hay un número definido de iteraciones. Debe decidirlo el equipo.
- Más realista que los modelos clásicos.
- **Ventajas:**
  - Monitoriza y controla riesgos continuamente
  - Integra el desarrollo con el mantenimiento
- **Inconvenientes:**
  - Tiene una elevada complejidad. Es muy complicado planificar un proyecto con este modelo ya que hay mucha incertidumbre acerca de las iteraciones a dar.
  - Requiere profesionales de gran experiencia
  - Es un modelo costoso
  - Requiere metas claras

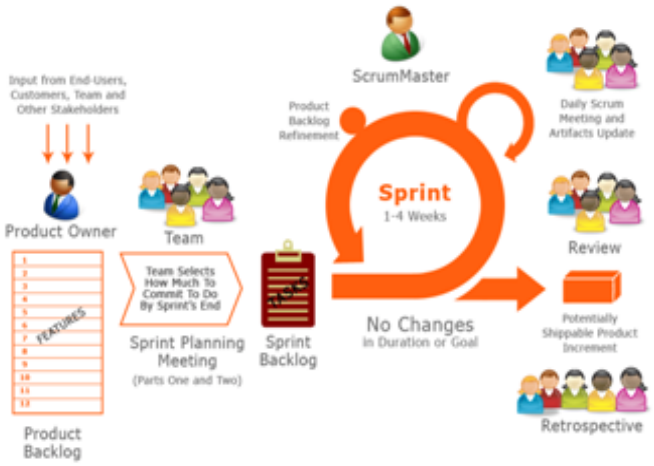


Metodologías ágiles

- El término ágil nace en el año 2001. Un grupo de expertos buscaba una alternativa a los procesos de desarrollo de software tradicionales (muy rígidos y dirigidos por la documentación).
- Se basan en el desarrollo iterativo e incremental
- Grupos auto organizados y multidisplinarios
- Buscan satisfacer al cliente mediante entrega de productos tempranas, funcionales y continuas
- Los cambios en los requisitos son requeridos
- Entregas frecuentes en el menor tiempo posible
- El equipo de desarrollo y el cliente deben trabajar juntos todo el proyecto
- Es más simple y aumenta la productividad
- **Ventajas**
  - Respuesta rápida a cambios de requisitos durante el proyecto
  - Mayor velocidad y eficiencia. Minimiza costos
  - Se identifican errores rápidamente debido a que se van haciendo pruebas a medida que se avanza
  - El equipo de desarrollo conoce el estado del proyecto en todo momento
  - Mejora la calidad del producto
- Algunos ejemplos son eXtreme Programming (XP), Scrum, Kanban, Open Up…

Metodologías ágiles – Scrum

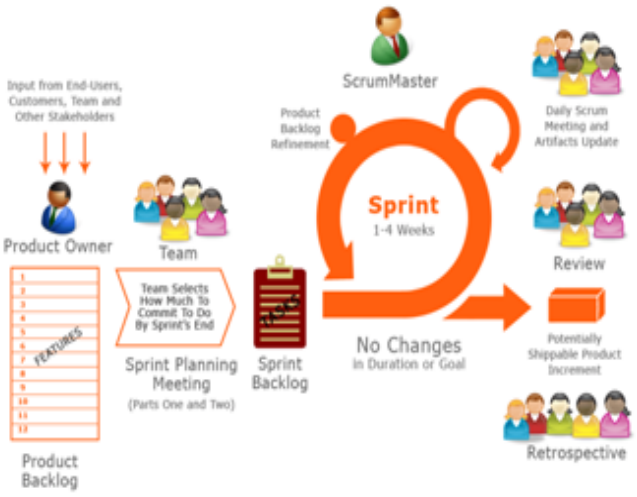
- Es una metodología ágil para la gestión de **todo tipo de proyectos**
- Los pilares de Scrum son:
  - **El ciclo de vida iterativo e incremental**
    - Va liberando partes poco a poco y cada entrega es un incremento de funcionalidad con respecto al anterior
    - Cada iteración se llama **sprint:** periodo de corta duración, menor de 4 semanas, que finaliza con un **prototipo operativo o producto potencialmente entregable**.
    - Lo que se implementa en cada sprint proviene de la **pila del producto (Product Backlog)** que contiene un conjunto de ítems



- Una figura clave es el propietario del producto (**Product Owner**): responsable de gestionar, mantener y priorizar el Product Backlog.
- Otra figura importante es el **Scrum Master:** una sola persona que ayuda al equipo y al Product Owner a finalizar con éxito, evitando incidentes, resolviendo cuellos de botella, etc. y haciendo que se cumpla el método Scrum
- Una vez seleccionado el ítem o ítems a desarrollar en el sprint el equipo los divide en la **pila del sprint (Sprint Backlog), que será inamovible durante el sprint.**

Metodologías ágiles – Scrum

- **Reuniones a lo largo del proyecto**
  - Se logra transparencia y comunicación y hacen que el equipo sea auto-gestionado y multifuncional
  - **Reunión de Planificación del Sprint (Sprint Planning Meeting)**: al principio de cada Sprint, para decidir que se va a realizar en ese Sprint.
  - **Reunión diaria (Daily Scrum)**: máximo 15 minutos, en la que se trata qué hizo ayer, qué va a hacer hoy y qué problemas se han encontrado.
  - **Reunión de Revisión del Sprint (Sprint Review Meeting)**: al final de cada Sprint, y se trata qué ha completado y qué no. También se muestra el trabajo al Product Owner.



- **Retrospectiva del Sprint (Sprint Retrospective)**: también al final del Sprint, y sirve para que los implicados den sus impresiones sobre el Sprint, y se utiliza para la mejora del proceso.