Curso 2023/24

# Tema-01 Parte II (Cont. 2ªEval)

## Introducción al desarrollo del software: Hardware y Software.

Entornos de desarrollo – IES BARAJAS
MIGUEL HH

# Tema-01(Cont. 2ªEval) (Parte II)

## Contenido

# 5.- INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE:

❖ **¿Qué es la Ingeniería del Software?**:

La ingeniería de software es una disciplina formada por un conjunto de métodos, herramientas y técnicas que se utilizan en el desarrollo y producción de software y programas informáticos, desde las primeras etapas de especificación del sistema hasta el mantenimiento del sistema después de que éste se haya puesto en uso. Se preocupa de las teorías, métodos y herramientas para el desarrollo profesional de software. La ingeniería del software se preocupa del desarrollo de software rentable.

El concepto Ingeniería trasciende a, (va más allá de), la programación, que es la base para crear una aplicación. La ingeniería de software engloba toda la gestión de un proyecto. Desde el análisis previo de la situación, el planteamiento del diseño hasta su implementación, pasando por las pruebas recurrentes para su correcto funcionamiento. Podríamos decir que la ingeniería del software es el continente donde se aloja el contenido, que sería el software en sí.

La ingeniería del software debería adoptar un enfoque sistemático y organizado para su trabajo y usar las herramientas y técnicas apropiadas dependiendo del problema a solucionar, las restricciones de desarrollo y los recursos disponibles.

## ***Definiciones de Ingeniería de Software a lo largo del tiempo***

**NATO – 1968**

Enfoque sistemático, disciplinado y cuantificable del desarrollo, operación y mantenimiento de software.

**Parnas – 1978**

La construcción de múltiples versiones de un software llevada a cabo por múltiples personas.

**Ghezzi – 1991**

Construcción de software de una envergadura o complejidad tales que debe ser construido por equipos de ingenieros.

**Parnas – 1997**

Reemplazar las renuncias de responsabilidad por garantías

**Jackson – 1998**

La ingeniería tradicional es altamente especializada y se basa en colecciones de diseños estándar o normalizados. ¿Hay especialidades en la Informática o cualquiera hace cualquier cosa? ¿Se basa la producción de software en diseños estándar? ¿Puede?

En resumen, podemos definir ***La ingeniería del software*** como la aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software, que es la aplicación de la ingeniería del software (IEEE, 1990).

## ***En resumen***

La Ingeniería de Software es o debería ser:

- Desarrollo de software de dimensión industrial
- Desarrollo sistemático, disciplinado y cuantificable
- Desarrollo de productos que tienen una vida muy larga
- Desarrollo en equipo
- Especialización
- Diseños estándar
- Producir software garantizado

❖ **Etapas de la Ingeniería de Software**:

Dentro de la ingeniería de software entendemos que también se encuentra todo el proceso de elaboración del software, que se denomina ***ciclo de vida del software***. Está formado por ***cuatro etapas***:

***Concepción***: En esta primera fase se desarrolla el modelo de negocio. Es decir, conocemos las necesidades que quiere un usuario y que debe de tener un software y empezamos a buscar las herramientas para cubrirlas.

***Elaboración***: Se detallan las características de la estructura del software.

***Construcción***: Implementación.

Tal y como su nombre indica en este paso empezamos a elaborar de forma tangible todo aquello que, de momento, solo hemos plasmado en forma de ideas.

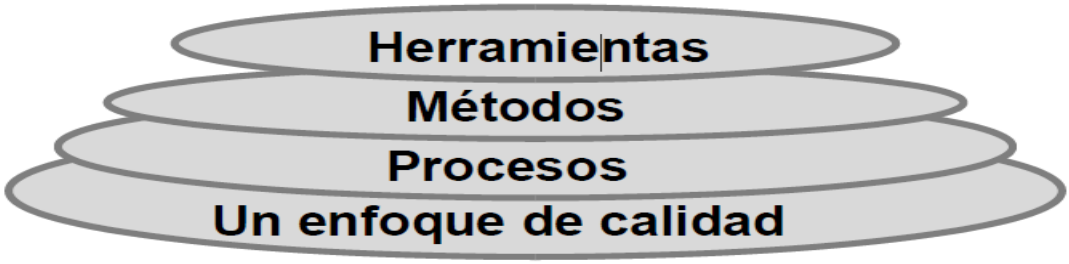
***Transición***: Es el momento de la implantación y el desarrollo para los clientes o usuarios. Deben tener tiempo para familiarizarse con el nuevo software.

***Una vez finalizado el ciclo de vida del software se realizan las siguientes tareas***:

***Mantenimiento***: Una vez se realiza todo el ciclo, entramos en otra fase conocida como mantenimiento. Es una de las etapas más importantes ya que se solucionan los problemas o errores que puedan surgir durante su implementación y también su posterior puesta en marcha (que se denomina Explotación). Además, se incorporan actualizaciones teniendo en cuenta los requisitos del cliente con el objetivo de que puedan cumplir la mayor cantidad de tareas.

## ❖ Capas de la Ingeniería del Software:

La ingeniería del software es una tecnología multicapa. Se puede ver como un conjunto de componentes estratificados, que reposan sobre ese enfoque de calidad.




Estos componentes que forman parte de la ingeniería del software son:

- **Herramientas**: la ayuda automatizada para los procesos y métodos.
- **Procesos**: un marco de trabajo que ayuda al jefe de proyecto a controlar la gestión del proyecto y las actividades de ingeniería.
- **Métodos**: las actividades técnicas requeridas para la creación de productos de trabajo.

***Herramientas***:

La capa de herramientas proporciona soporte a las capas de proceso y métodos centrándose en el significado de la automatización de algunas de las actividades manuales.

Las herramientas se pueden utilizar para automatizar las siguientes actividades:

- Actividades de gestión de proyectos.
- Métodos técnicos usados en la ingeniería del software.
- Soporte de sistemas general.
- Marcos de trabajo para otras herramientas.

La automatización ayuda a eliminar el tedio del trabajo, reduce las posibilidades de errores, y hace más fácil usar buenas prácticas de ingeniería del software. Cuando se usan herramientas, la documentación se convierte en una parte integral del trabajo hecho, en vez de ser una actividad adicional. De ahí que la documentación no se tenga que realizar como actividad adicional. Las herramientas se pueden utilizar para realizar actividades de gestión de proyecto, así como para actividades técnicas.

Existen una gran variedad de herramientas para múltiples actividades. Entre ellas se pueden destacar las siguientes:

- Herramientas de gestión de proyectos.
- Herramientas de control de cambios.
- Herramientas de análisis y diseño.
- Herramientas de generación de código.
- Herramientas de pruebas.
- Herramientas de reingeniería.

- Herramientas de documentación.
- Herramientas de prototipos.

Estas herramientas soportan las capas de proceso y de métodos en varias actividades.

***Procesos***:

El fundamento de la ingeniería del software es la capa de proceso. El proceso define un marco de trabajo para un conjunto de áreas clave de proceso que se deben establecer para la entrega efectiva de la tecnología de la ingeniería del software.

La capa de proceso define el proceso que se usará para construir el software y las actividades y tareas que un jefe de proyecto tiene que gestionar. Por lo tanto, las áreas claves del proceso forman la base del control de gestión de proyectos del software y establecen el contexto en el que se aplican los métodos técnicos, se obtienen productos de trabajo (modelos, documentos, datos, informes, formularios, etc.), se establecen hitos, se asegura la calidad y el cambio se gestiona adecuadamente. El proceso de la ingeniería del software es la unión que mantiene juntas las capas de tecnologías y que permite un desarrollo racional y oportuno de la ingeniería del software.

Se pueden ver todas las actividades, incluyendo las actividades técnicas, como parte del proceso. Además, cualquier recurso, incluyendo herramientas usadas para construir el software también encajan en el proceso. La capa de proceso es, por lo tanto, el fundamento de la ingeniería del software y da soporte a las capas de métodos y herramientas.

Todos los enfoques de la construcción de software tienen un proceso, pero en muchos casos, son ad hoc, invisibles y caóticos. Una buena ingeniería de software hace que el proceso de software sea más visible, predecible y más útil para aquellos que construyen software.

***Métodos***:

La capa de métodos se centra en las actividades técnicas que se deben realizar para conseguir las tareas de ingeniería. Proporciona el “cómo” y cubre las actividades de ingeniería fundamentales.

Los métodos abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento. Los métodos de la ingeniería del software dependen de un conjunto de principios básicos que gobiernan cada una de las áreas de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas.

La construcción de software implica una amplia colección de actividades técnicas. La capa de métodos contiene los métodos definidos para realizar esas actividades de forma eficiente. Se centra en cómo se han de realizar las actividades técnicas. Las personas involucradas usan los métodos para realizar las actividades de ingeniería fundamentales necesarias para construir el software.

Para varias actividades de proceso, la capa de métodos contiene el correspondiente conjunto de métodos técnicos para usar. Esto abarca un conjunto de reglas, los modos de representación gráficos o basados en texto, y las guías relacionadas para la evaluación de la calidad de la información representada.

Para definir la capa de métodos, es necesario seleccionar un método adecuado de un amplio rango de métodos disponibles.

## ❖ Objetivos de la Ingeniería de Software:

La ingeniería de software cubre un marco muy amplio. Hay que entender esto como la posibilidad de que enmarque varios objetivos a tener en cuenta cuando queremos implementar u optar por un servicio de ingeniería de software:

- Diseño de programas informáticos adaptados a las necesidades y exigencias de los clientes.
- Solucionar problemas de programación.
- Estar presente en todas las fases del ciclo de vida de un producto.
- Contabilizar los costes de un proyecto y evaluar los tiempos de desarrollo.
- Realizar el seguimiento del presupuesto y cumplir los plazos de entrega.
- Liderar equipos de trabajo de desarrollo de software.
- Estructurar la elaboración de evidencias que comprueben el perfecto funcionamiento de los programas y que se adaptan a los requerimientos de análisis y diseño.
- Diseñar, construir y administrar bases de datos.
- Liderar y orientar a los programadores durante el desarrollo de aplicaciones.
- Incluir procesos de calidad en los sistemas, calculando métricas e indicadores y chequeando la calidad del software producido.
- Estructurar e inspeccionar el trabajo del equipo ya sea el grupo de técnicos de mantenimiento o el grupo de ingenieros de sistemas y redes.

No siempre una ingeniería de software debe enfocarse a todos estos objetivos. Es decir, se dirigen hacia la consecución de algunos de ellos, pero no necesariamente a todos, ya que las empresas que contratan este servicio no requieren el mismo tipo de proyecto.

## ❖ Arquitectura de sistemas:

Relacionado con la ingeniería de software también se encuentra la ***Arquitectura de sistemas***. Consiste en la esquematización de la estructura general del proyecto a desarrollar. El objetivo de conocer este algo más que el esqueleto del software es tener la capacidad de señalar y conocer cuáles son los componentes que son necesarios para llevar a cabo el desarrollo.

Hay que tener en cuenta que ***existen dos tipos de softwares***. Por un lado, destacamos ***el estándar***, más generalista y que se puede adaptar a varios modelos de negocio. Mientras que, por el otro lado, tenemos ***el personalizado***. Se trata de un tipo de software que se desarrolla para el uso exclusivo de un cliente. Se diseña a su imagen y semejanza, por lo que es lógico que solamente sirve para esa empresa, ya que se adapta a las necesidades y características de la compañía que ha solicitado diseñarlo. A pesar de estas diferencias clave todos los softwares presentan tres elementos que lo caracterizan: Programas y/o algoritmos, Estructura de datos y Documentos.

# 6.- CICLO DE VIDA DE DESARROLLO DEL SOFTWARE:

El ciclo de vida es el conjunto de fases por las que pasa el sistema que se está desarrollando desde que nace la idea inicial hasta que el software es retirado o remplazado (muere). También se denomina a veces paradigma.

Entre las funciones que debe tener un ciclo de vida se pueden destacar:

- Determinar el orden de las fases del proceso de software.
- Establecer los criterios de transición para pasar de una fase a la siguiente.
- Definir las entradas y salidas de cada fase.
- Describir los estados por los que pasa el producto.
- Describir las actividades a realizar para transformar el producto.
- Definir un esquema que sirve como base para planificar, organizar, coordinar, desarrollar… .

Un ciclo de vida para un proyecto se compone de fases sucesivas compuestas por tareas que se pueden planificar. Según el modelo de ciclo de vida, la sucesión de fases puede ampliarse con bucles de realimentación, de manera que lo que conceptualmente se considera una misma fase se pueda ejecutar más de una vez a lo largo de un proyecto, recibiendo en cada pasada de ejecución aportaciones a los resultados intermedios que se van produciendo (realimentación).

**Fases**:

Una fase es un conjunto de actividades relacionadas con un objetivo en el desarrollo del proyecto. Se construye agrupando tareas (actividades elementales) que pueden compartir un tramo determinado del tiempo de vida de un proyecto. La agrupación temporal de tareas impone requisitos temporales correspondientes a la asignación de recursos (humanos, financieros o materiales).

**Entregables**:

Son los productos intermedios que generan las fases. Pueden ser materiales o inmateriales (documentos, software). Los entregables permiten evaluar la marcha del proyecto mediante comprobaciones de su adecuación o no a los requisitos funcionales y de condiciones de realización previamente establecidos.

Las actividades genéricas del ciclo de vida del desarrollo del software son:

- **Especificación**: Lo que el sistema debería hacer y sus restricciones de desarrollo.
- **Desarrollo**: Producción del sistema software.
- **Validación**: Comprobar que el sistema es lo que el cliente quiere.
- **Evolución**: Cambiar el software en respuesta a las demandas de cambio.

# 7.- MODELOS DE CICLO DE VIDA DEL SOFTWARE:

La ingeniería del software se vale de una serie de modelos que establecen y muestran las distintas etapas y estados por los que pasa un producto software, desde su concepción inicial, pasando por su desarrollo, puesta en marcha y posterior mantenimiento, hasta la retirada del producto. A estos modelos se les denomina "Modelos de ciclo de vida del software". El primer modelo concebido fue el de Royce, más comúnmente conocido como "Cascada" o "Lineal Secuencial". Este modelo establece que las diversas actividades que se van realizando al desarrollar un producto software, se suceden de forma lineal.

Los modelos de ciclo de vida del software describen las fases del ciclo de software y el orden en que se ejecutan las fases.

Un modelo de ciclo de vida de software es una vista de las actividades que ocurren durante el desarrollo de software, intenta determinar el orden de las etapas involucradas y los criterios de transición asociados entre estas etapas.

***Un modelo de ciclo de vida del software***:

- Describe las fases principales de desarrollo de software.
- Define las fases primarias esperadas de ser ejecutadas durante esas fases.
- Ayuda a administrar el progreso del desarrollo.
- Provee un espacio de trabajo para la definición de un proceso detallado de desarrollo de software.

***Las principales diferencias entre distintos modelos de ciclo de vida están en:***

- El alcance del ciclo dependiendo de hasta dónde llegue el proyecto correspondiente. Un proyecto puede comprender un simple estudio de viabilidad del desarrollo de un producto, o su desarrollo completo o en el extremo, toda la historia del producto con su desarrollo, fabricación y modificaciones posteriores hasta su retirada del mercado.
- Las características (contenidos) de las fases en que dividen el ciclo. Esto puede depender del propio tema al que se refiere el proyecto, o de la organización.
- La estructura y la sucesión de las etapas, si hay realimentación entre ellas, y si tenemos libertad de repetirlas (iterar).

## ❖ **Modelo en cascada**:

Es un enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de forma que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.

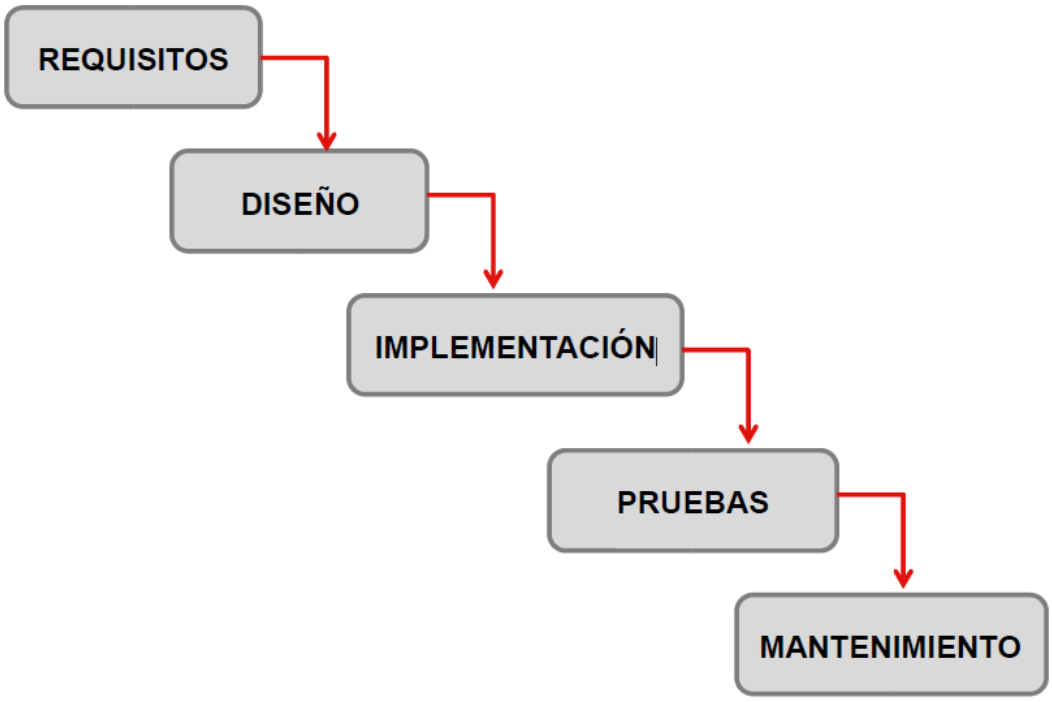
El modelo en cascada es un proceso de desarrollo secuencial, en el que el desarrollo se ve fluyendo hacia abajo (como una cascada) sobre las fases que componen el ciclo de vida.

La primera descripción formal del modelo en cascada se cree que fue en un artículo publicado en 1970 por Winston W. Royce, aunque Royce no usó el término cascada en este artículo. Irónicamente, Royce estaba presentando este modelo como un ejemplo de modelo que no funcionaba, defectuoso.

En el modelo original de Royce, existían las siguientes fases:

1. Especificación de requisitos.
2. Diseño.
3. Construcción (Implementación o codificación).
4. Integración.
5. Pruebas.
6. Instalación.
7. Mantenimiento.

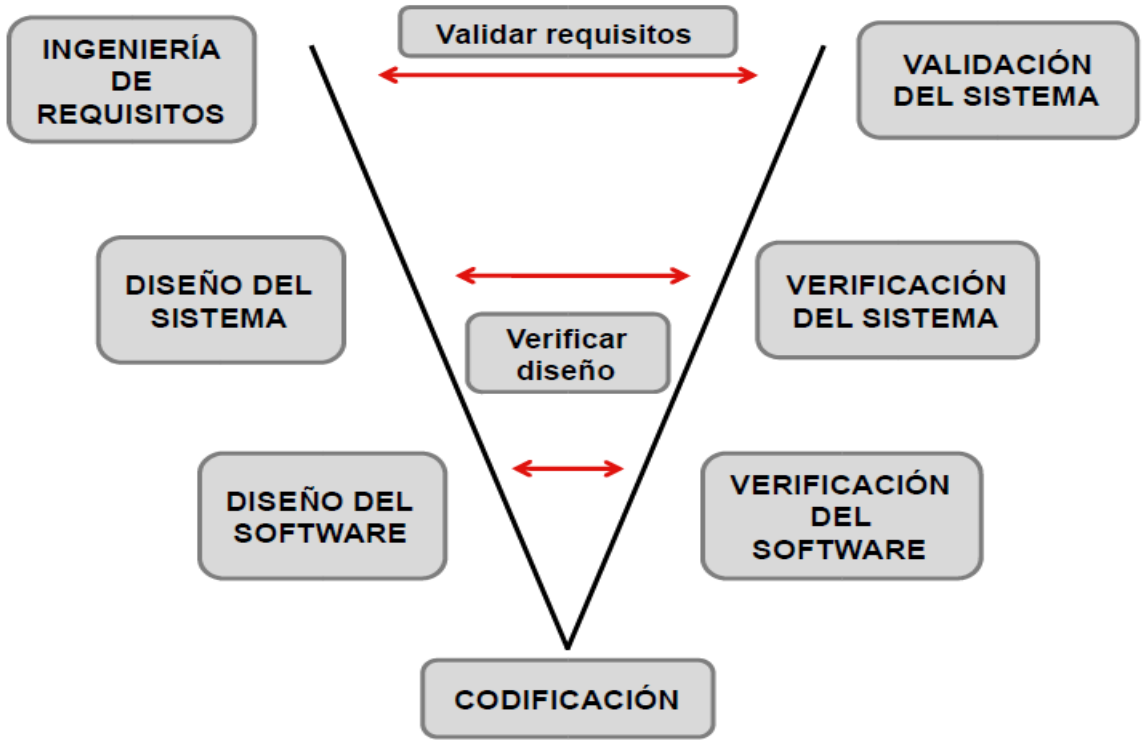
Para seguir el modelo en cascada, se avanza de una fase a la siguiente en una forma puramente secuencial. Si bien ha sido ampliamente criticado desde el ámbito académico y la industria, sigue siendo el paradigma más seguido a día de hoy. Representación del modelo de ciclo de vida en cascada:



## ❖ Modelo en V:

El modelo en v se desarrolló para terminar con algunos de los problemas que se vieron utilizando el enfoque de cascada tradicional. Los defectos estaban siendo encontrados demasiado tarde en el ciclo de vida, ya que las pruebas no se introducían hasta el final del proyecto. El modelo en v dice que las pruebas necesitan empezarse lo más pronto posible en el ciclo de vida. También muestra que las pruebas no son sólo una actividad basada en la ejecución. Hay una variedad de actividades que se han de realizar antes del fin de la fase de codificación. Estas actividades deberían ser llevadas a cabo en paralelo con las actividades de desarrollo, y los técnicos de pruebas necesitan trabajar con los desarrolladores y analistas de negocio de tal forma que puedan realizar estas actividades y tareas y producir una serie de entregables de pruebas.

El modelo en v es un proceso que representa la secuencia de pasos en el desarrollo del ciclo de vida de un proyecto. Describe las actividades y resultados que han de ser producidos durante el desarrollo del producto. La parte izquierda de la v representa la descomposición de los requisitos y la creación de las especificaciones del sistema. El lado derecho de la v representa la integración de partes y su verificación. V significa “Validación y Verificación”. Representación del modelo de ciclo de vida en V:

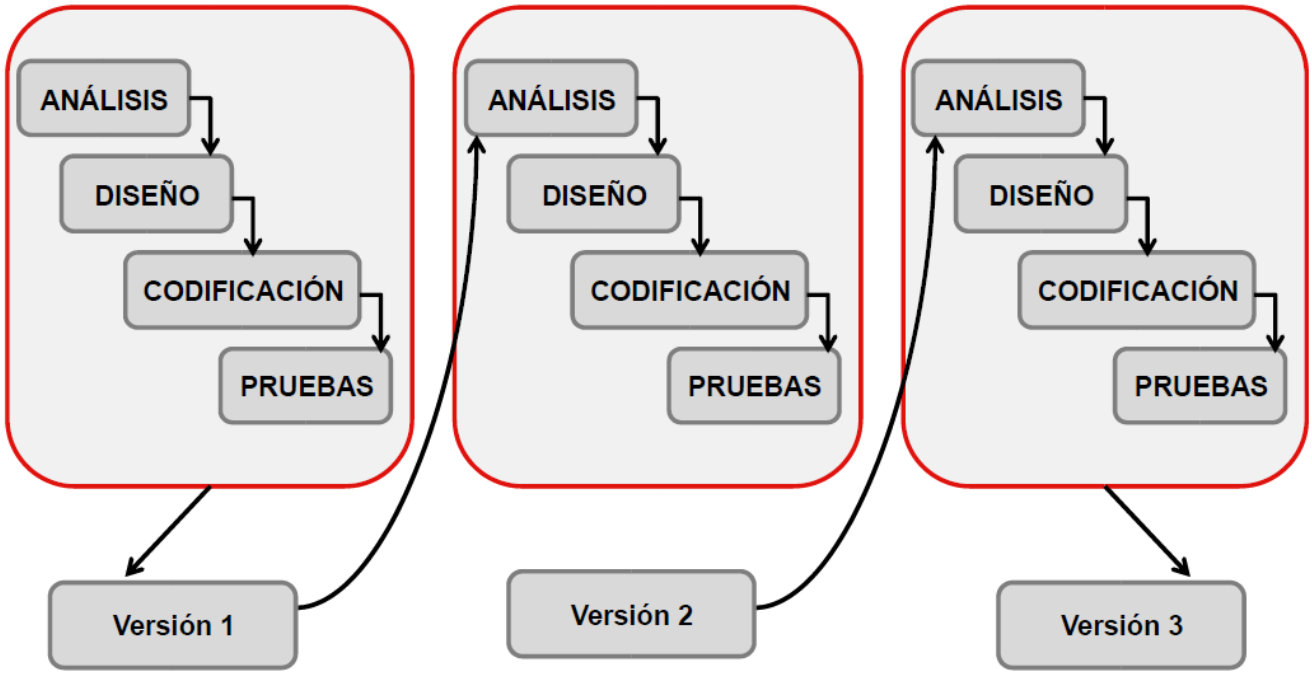



Realmente las etapas individuales del proceso pueden ser casi las mismas que las del modelo en cascada. Sin embargo, hay una gran diferencia. En vez de ir para abajo de una forma lineal las fases del proceso vuelven hacia arriba tras la fase de codificación, formando una v. La razón de esto es que para cada una de las fases de diseño se ha encontrado que hay un homólogo en las fases de pruebas que se correlacionan.

❖ **Modelo iterativo**:

Es un modelo derivado del ciclo de vida en cascada. Este modelo busca reducir el riesgo que surge entre las necesidades del usuario y el producto final por malos entendidos durante la etapa de recogida de requisitos.

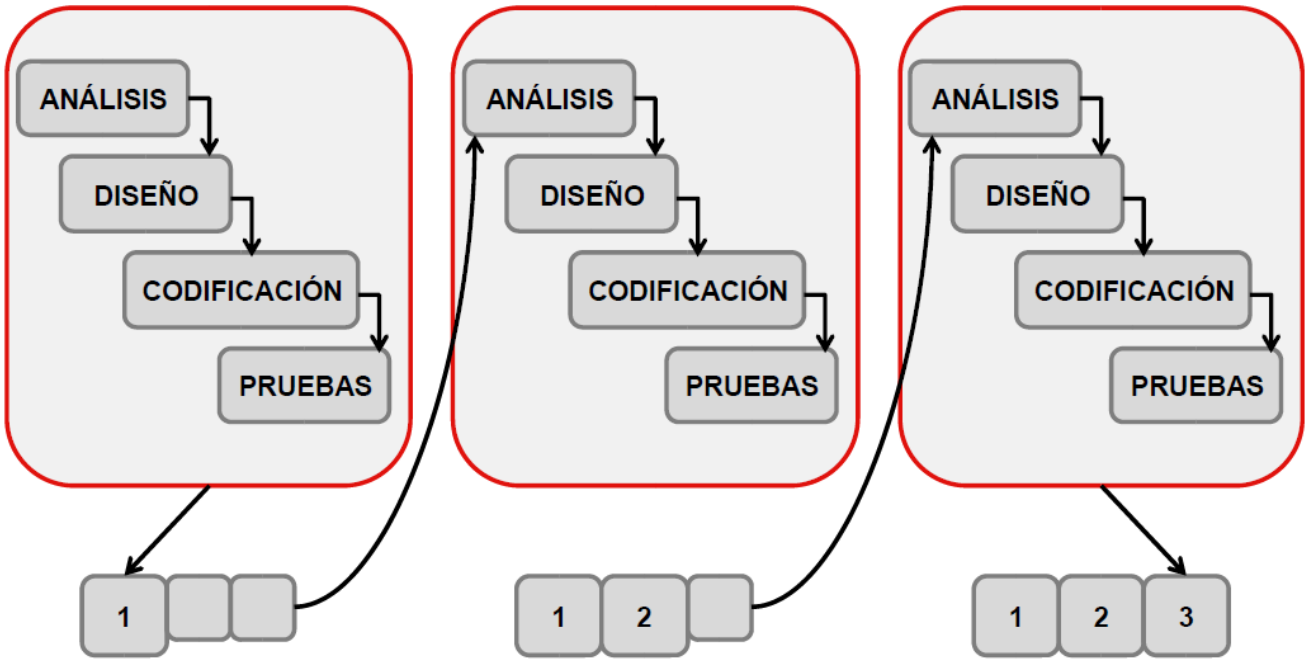
Consiste en la iteración de varios ciclos de vida en cascada. Al final de cada iteración se le entrega al cliente una versión mejorada o con mayores funcionalidades del producto. El cliente es quien, después de cada iteración, evalúa el producto y lo corrige o propone mejoras. Estas iteraciones se repetirán hasta obtener un producto que satisfaga las necesidades del cliente. Representación del modelo de ciclo de vida en iterativo:




Este modelo se suele utilizar en proyectos en los que los requisitos no están claros por parte del usuario, por lo que se hace necesaria la creación de distintos prototipos para presentarlos y conseguir la conformidad del cliente.

### ❖ **Modelo de desarrollo incremental**:

El modelo incremental combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos. Se basa en la filosofía de construir incrementando las funcionalidades del programa. Este modelo aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal produce un incremento del software. Representación del modelo de ciclo de vida en desarrollo incremental:




Cuando se utiliza un modelo incremental, el primer incremento es a menudo un producto esencial, sólo con los requisitos básicos. Este modelo se centra en la entrega de un producto operativo con cada incremento. Los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación.

## ❖ **Modelo en espiral**:

El desarrollo en espiral es un modelo de ciclo de vida desarrollado por Barry Boehm en 1985, utilizado de forma generalizada en la ingeniería del software. Las actividades de este modelo se conforman en una espiral, cada bucle representa un conjunto de actividades. Las actividades no están fijadas a priori, sino que las siguientes se eligen en función del análisis de riesgos, comenzando por el bucle anterior.

Al ser un modelo de ciclo de vida orientado a la gestión de riesgos se dice que uno de los aspectos fundamentales de su éxito radica en que el equipo que lo aplique tenga la necesaria experiencia y habilidad para detectar y catalogar correctamente riesgos.

### **Tareas:**

*Para cada ciclo habrá cuatro actividades:*

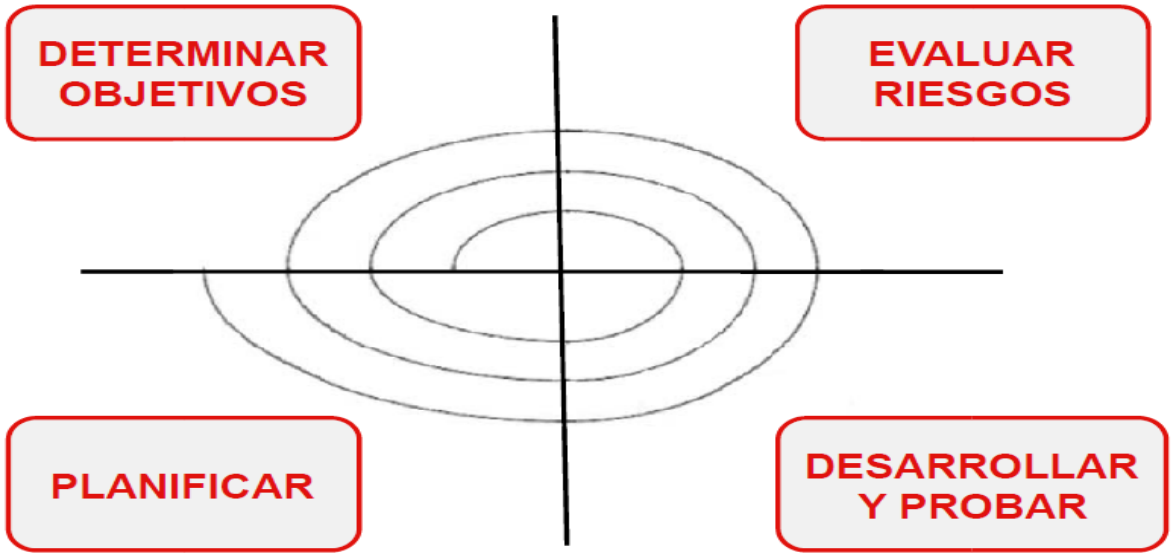
I. Determinar o fijar objetivos:
   - Fijar también los productos definidos a obtener: requerimientos, especificación, manual de usuario.
   - Fijar las restricciones.
   - Identificar riesgos del proyecto y estrategias alternativas para evitarlos.
   - Hay una cosa que solo se hace una vez: planificación inicial o previa.

II. Análisis del riesgo:
   - Estudiar todos los riesgos potenciales y se seleccionan una o varias alternativas propuestas para reducir o eliminar los riesgos.

III. Desarrollar, verificar y validar (probar):
   - Tareas de la actividad propia y de prueba.
   - Análisis de alternativas e identificación de resolución de riesgos.
   - Dependiendo del resultado de la evaluación de riesgos, se elige un modelo para el desarrollo, que puede ser cualquiera de los otros existentes, como formal, evolutivo, cascada, etc. Así, por ejemplo, si los riesgos de la interfaz de usuario son dominantes, un modelo de desarrollo apropiado podría ser la construcción de prototipos evolutivos.

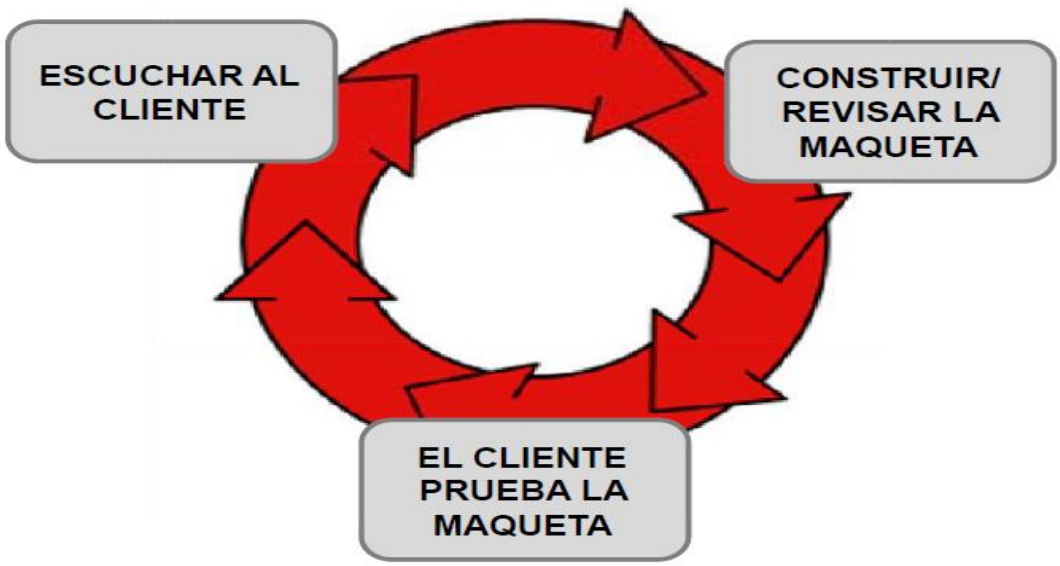
IV. Planificar:
   - Revisar todo lo que se ha llevado a cabo, evaluándolo y decidiendo si se continua con las fases siguientes y planificando la próxima actividad.

El proceso empieza en la posición central. Desde allí se mueve en el sentido de las agujas del reloj. Representación del modelo de ciclo de vida en espiral:




❖ **Modelo de prototipos**:

El paradigma de construcción de prototipos comienza con la recolección de requisitos. El desarrollador y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos y las áreas del esquema en donde es obligatoria más definición. Entonces aparece un diseño rápido. El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente. El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer. Representación del modelo de ciclo de vida en prototipos:



❖ **Comparativa de los modelos de ciclo de vida**:

A continuación, se van a mostrar en una tabla, las ventajas e inconvenientes más significativas de cada uno de los modelos de ciclo de vida:

| MODELOS | VENTAJAS | INCONVENIENTES |
|---|---|---|
| **Modelo en cascada** | ▪ Modelo en el que está todo bien organizado.<br>▪ No se mezclan las fases.<br>▪ Simple y fácil de llevar a la práctica.<br>▪ Fácil de gestionar. | ▪ Rara vez los proyectos siguen una secuencia lineal.<br>▪ Difícil establecer todos los requisitos al principio.<br>▪ Visibilidad del producto cuando está terminado. |
| **Modelo en V** | ▪ Simple y fácil de llevar a la práctica.<br>▪ En cada una de las fases hay entregables específicos.<br>▪ Desarrollo de planes de prueba en etapas tempranas del ciclo de vida.<br>▪ Suele funcionar bien para proyectos pequeños donde los requisitos son entendidos fácilmente. | ▪ Tiene poca flexibilidad y ajustar el alcance es difícil y caro.<br>▪ El modelo no proporciona caminos claros para problemas encontrados durante las fases de pruebas. |
| **Modelo incremental** | ▪ Se genera software operativo de forma rápida y en etapas tempranas del ciclo de vida del software.<br>▪ Modelo más flexible, por lo que se reduce el coste en cambios de alcance y requisitos.<br>▪ Es más fácil probar y depurar en una iteración más pequeña.<br>▪ Es más fácil gestionar riesgos.<br>▪ Cada iteración es un hito gestionado fácilmente. | ▪ Se requiere mucha experiencia para definir los incrementos y distribuir en ellos las tareas de forma proporcionada.<br>▪ Cada fase de una iteración es rígida y no se superpone con otras.<br>▪ Todos los requisitos han de definirse al inicio. |
| **Modelo iterativo** | ▪ No hace falta que los requisitos estén totalmente definidos desde el principio.<br>▪ Desarrollo en pequeños ciclos.<br>▪ Es más fácil gestionar riesgos.<br>▪ Cada iteración es un hito gestionado fácilmente. | ▪ Que los requisitos no estén definidos desde el principio también puede verse como un inconveniente ya que pueden surgir problemas con la arquitectura. |

| | | |
|---|---|---|
| **Modelo de prototipos** | ▪ Visibilidad del producto desde el inicio del ciclo de vida con el primer prototipo.<br>▪ Permite introducir cambios en las iteraciones siguientes del ciclo.<br>▪ Permite la realimentación continua del cliente. | ▪ Puede ser un desarrollo lento. |
| **Modelo en espiral** | ▪ Reduce riesgos del proyecto.<br>▪ Incorpora objetivos de calidad.<br>▪ Integra el desarrollo con el mantenimiento.<br>▪ No es rígido ni estático.<br>▪ Se produce software en etapas tempranas del ciclo de vida. | ▪ Modelo que genera mucho trabajo adicional.<br>▪ Exige un alto nivel de experiencia y cierta habilidad en los analistas de riesgos.<br>▪ Modelo costoso. |

### ❖ Normativa estándar ISO/IEC 12207 para el ciclo de vida del software:

Esta norma establece un marco de referencia común para los procesos del ciclo de vida del software, con una terminología bien definida a la que puede hacer referencia la industria del software. Contiene procesos, actividades y tareas para aplicar durante la adquisición de un sistema que contiene software, un producto software puro o un servicio software, y durante el suministro, desarrollo, operación y mantenimiento de productos software. El software incluye la parte software del firmware.

Esta norma incluye también un proceso que puede emplearse para definir, controlar y mejorar los procesos del ciclo de vida del software.

La ***ISO 12207*** define un modelo de ciclo de vida como un marco de referencia que contiene los procesos, actividades y tareas involucradas en el desarrollo, operación y mantenimiento de un producto software, y que abarca toda la vida del sistema, desde la definición de sus requisitos hasta el final del uso.

Esta norma agrupa las actividades que pueden llevarse a cabo durante el ciclo de vida del software en cinco procesos principales, ocho procesos de apoyo y cuatro procesos organizativos. Cada proceso del ciclo de vida está dividido en un conjunto de actividades; cada actividad se subdivide a su vez en un conjunto de tareas.

- **Procesos principales del ciclo de vida**:
  Son cinco procesos que dan servicio a las partes principales durante el ciclo de vida del software. Una parte principal es la que inicia o lleva a cabo el desarrollo, operación y mantenimiento de productos software. Los procesos principales son:
  - Proceso de adquisición.
  - Proceso de suministro.
  - Proceso de desarrollo.
  - Proceso de operación.
  - Proceso de mantenimiento

- **Procesos de apoyo al ciclo de vida**:
  Son procesos que apoyan a otros procesos como parte esencial de los mismos, con un propósito bien definido, y contribuyen al éxito y calidad del proyecto software. Un proceso de apoyo se emplea y ejecuta por otro proceso según sus necesidades.
  Los procesos de apoyo son:
  - Proceso de documentación.
  - Proceso de gestión de la configuración.
  - Proceso de verificación.
  - Proceso de validación.
  - Proceso de revisiones conjuntas.
  - Proceso de auditoría.
  - Proceso de solución de problemas.

- **Procesos organizativos del ciclo de vida**:
  Se emplean por una organización para establecer e implementar una infraestructura construida por procesos y personal asociado al ciclo de vida, y para mejorar continuamente esta estructura y procesos.
  - Proceso de gestión.
  - Proceso de infraestructura.
  - Proceso de mejora.
  - Proceso de formación.

# 8.- METODOLOGÍAS DE DESARROLLO DE SOFTWARE:

Un objetivo de décadas ha sido encontrar procesos y metodologías que sean sistemáticas, predecibles y repetibles, a fin de mejorar la productividad en el desarrollo y la calidad del producto software. Existen numerosas propuestas ***metodológicas*** que inciden en distintas dimensiones del proceso de desarrollo. Por una parte, tenemos aquellas propuestas más ***tradicionales*** que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en muchos otros. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de las ***metodologías ágiles***, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo, pero manteniendo una alta calidad. Las metodologías ágiles están revolucionando la manera de producir software, y a la vez generando un amplio debate entre sus seguidores y quienes por escepticismo o convencimiento no las ven como alternativa para las metodologías tradicionales.

La evolución de la disciplina de ingeniería del software ha traído consigo propuestas diferentes para mejorar los resultados del proceso de construcción. Las metodologías tradicionales haciendo énfasis en la planificación y las metodologías ágiles haciendo énfasis en la adaptabilidad del proceso, delinean las principales propuestas presentes.

### ❖ **Definición de metodología**:

*Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo*.

Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, incremental…). Definen artefactos, roles y actividades, junto con prácticas y técnicas recomendadas.

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo por el cual fue creado.

Una definición estándar de metodología puede ser el conjunto de métodos que se utilizan en una determinada actividad con el fin de formalizarla y optimizarla. Determina los pasos a seguir y cómo realizarlos para finalizar una tarea.

Si esto se aplica a la ingeniería del software, podemos destacar que una metodología:

- Optimiza el proceso y el producto software.
- Proporciona métodos que guían en la planificación y en el desarrollo del software.
- Define qué hacer, cómo y cuándo durante todo el desarrollo y mantenimiento de un proyecto.

Una metodología define una estrategia global para enfrentarse con el proyecto. Entre los elementos que forman parte de una metodología se pueden destacar:

- **Fases:** Tareas a realizar en cada fase.
- **Productos:** E/S de cada fase, documentos.
- **Procedimientos y herramientas:** Apoyo a la realización de cada tarea.
- **Criterios de evaluación del proceso y del producto:** Permiten determinar si se han logrado los objetivos

El marco de trabajo de una metodología de desarrollo de software consiste en:

- Una filosofía de desarrollo de software, con el enfoque o enfoques del proceso de desarrollo de software.
- Múltiples herramientas, modelos y métodos para ayudar en el proceso de desarrollo de software.

Estos marcos de trabajo están con frecuencia vinculados a algunos tipos de organizaciones, que se encargan del desarrollo, soporte de uso y promoción de la metodología. La metodología con frecuencia se documenta de alguna manera formal.

## ❖ Ventajas del uso de una metodología:

Son muchas las ventajas que puede aportar el uso de una metodología. A continuación, se van a exponer algunas de ellas, clasificadas desde distintos puntos de vista:

***Desde el punto de vista de gestión***.

- Facilitar la tarea de planificación.
- Facilitar la tarea del control y seguimiento de un proyecto.
- Mejorar la relación coste/beneficio.
- Optimizar el uso de los recursos disponibles.
- Facilitar la evaluación de resultados y el cumplimiento de los objetivos.
- Facilitar la comunicación efectiva entre usuarios y desarrolladores

***Desde el punto de vista de los ingenieros del software:***

- Ayudar a la comprensión del problema.
- Optimizar el conjunto y cada una de las fases del proceso de desarrollo.
- Facilitar el mantenimiento del producto final.
- Permitir la reutilización de partes del producto.

***Desde el punto de vista del cliente o usuario:***

- Garantizar un determinado nivel de calidad en el producto final.
- Ofrecer confianza en los plazos de tiempo fijados en la definición del proyecto.
- Definir el ciclo de vida que más se adecue a las condiciones y características del desarrollo.

## ❖ Metodologías tradicionales y ágiles:

Desarrollar un buen software depende de un gran número de actividades y etapas, donde el impacto de elegir la metodología para un equipo en un determinado proyecto es trascendental para el éxito del producto.

Según la filosofía de desarrollo se pueden clasificar las metodologías en dos grupos. Las metodologías tradicionales, que se basan en una fuerte planificación durante todo el desarrollo, y las metodologías ágiles, en las que el desarrollo de software es incremental, cooperativo, sencillo y adaptado.

**Metodologías tradicionales**

Las metodologías tradicionales son denominadas, a veces, de forma peyorativa, como metodologías pesadas.

Centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto.

Otra de las características importantes dentro de este enfoque, son los altos costes al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil.

Las metodologías tradicionales (formales) se focalizan en la documentación, planificación y procesos (plantillas, técnicas de administración, revisiones, etc.)

**Metodologías ágiles**

Este enfoque nace como respuesta a los problemas que puedan ocasionar las metodologías tradicionales y se basa en dos aspectos fundamentales, retrasar las decisiones y la planificación adaptativa. Basan su fundamento en la adaptabilidad de los procesos de desarrollo.

Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan.

- **¿Metodologías ágiles o metodologías tradicionales?**:

En las metodologías tradicionales el principal problema es que nunca se logra planificar bien el esfuerzo requerido para seguir la metodología. Pero entonces, si logramos definir.

métricas que apoyen la estimación de las actividades de desarrollo, muchas prácticas de metodologías tradicionales podrían ser apropiadas. El no poder predecir siempre los resultados de cada proceso no significa que estemos frente a una disciplina de azar. Lo que significa es que estamos frente a la necesidad de adaptación de los procesos de desarrollo que son llevados por parte de los equipos que desarrollan software.

Tener metodologías diferentes para aplicar de acuerdo con el proyecto que se desarrolle resulta una idea interesante. Estas metodologías pueden involucrar prácticas tanto de metodologías ágiles como de metodologías tradicionales. De esta manera podríamos tener una metodología por cada proyecto, la problemática sería definir cada una de las prácticas, y en el momento preciso definir parámetros para saber cuál usar.

Es importante tener en cuenta que el uso de un método ágil no vale para cualquier proyecto. Sin embargo, una de las principales ventajas de los métodos ágiles es su peso inicialmente ligero y por eso las personas que no estén acostumbradas a seguir procesos encuentran estas metodologías bastante agradables.

A continuación, se va a mostrar una tabla que aparece una comparativa entre estos dos grupos de metodologías.

| METODOLOGÍAS ÁGILES | METODOLOGÍAS TRADICIONALES |
|---|---|
| Basadas en heurísticas provenientes de prácticas de producción de código | Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo |
| Especialmente preparados para cambios durante el proyecto | Cierta resistencia a los cambios |
| Impuestas internamente (por el equipo) | Impuestas externamente |
| Proceso menos controlado, con pocos principios | Proceso mucho más controlado, con numerosas políticas/normas |
| No existe contrato tradicional o al menos es bastante flexible | Existe un contrato prefijado |
| El cliente es parte del equipo de desarrollo | El cliente interactúa con el equipo de desarrollo mediante reuniones |
| Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio | Grupos grandes y posiblemente distribuidos |
| Pocos artefactos | Más artefactos |
| Pocos roles | Más roles |
| Menos énfasis en la arquitectura del software | La arquitectura del software es esencial y se expresa mediante modelos |

## 9.- BIBLIOGRAFÍA:


- Temario años anteriores del módulo *Entornos de desarrollo*.
- Agile Spain www.agile-spain.com
- Alianza ágil www.agilealliance.org
- IEEE www.ieee.org/portal/site
- Manifiesto ágil www.agilemanifesto.org
- Sitio web de la Organización Internacional para la Estandarización www.iso.org
- Swebok www.swebok.org