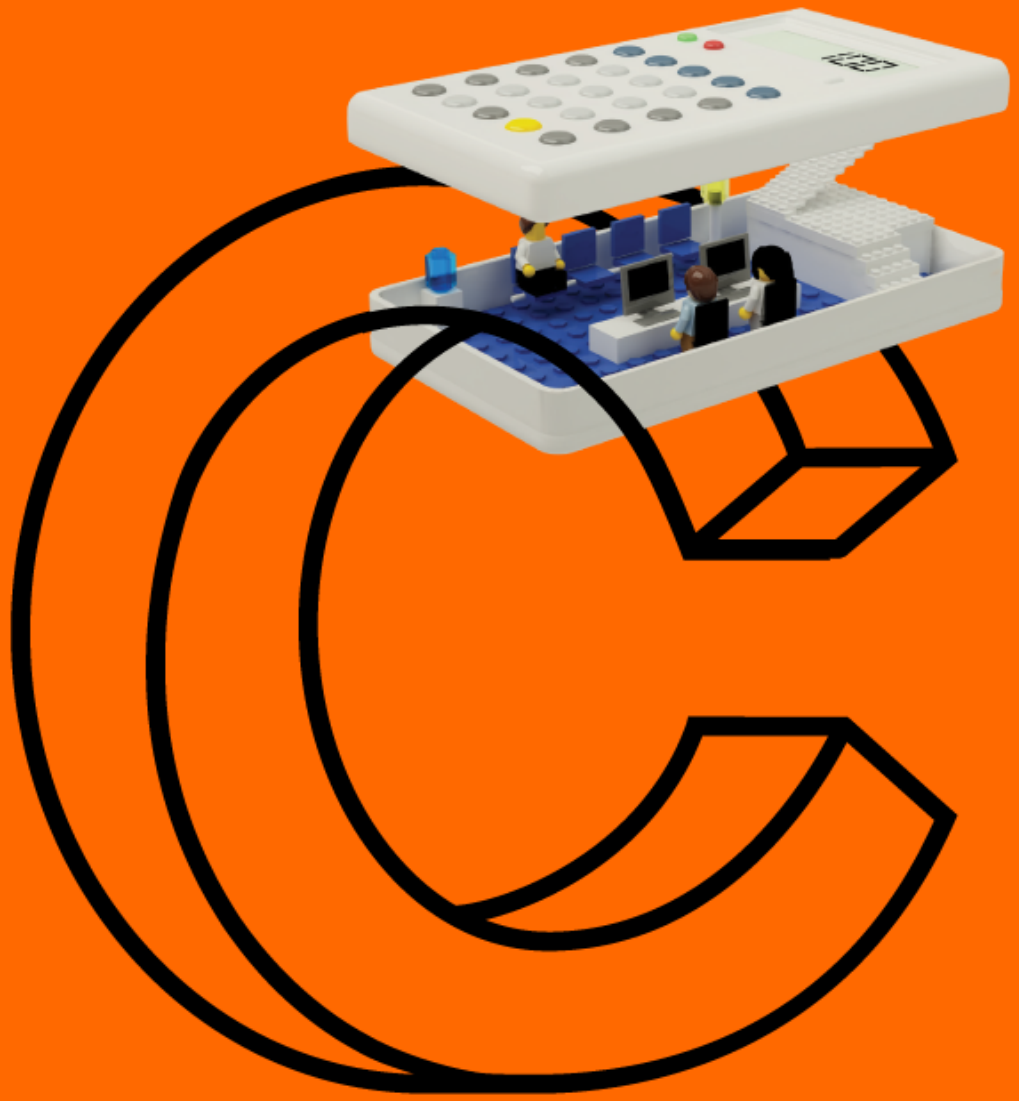
Análisis y Diseño de Sistemas Orientado a Objetos

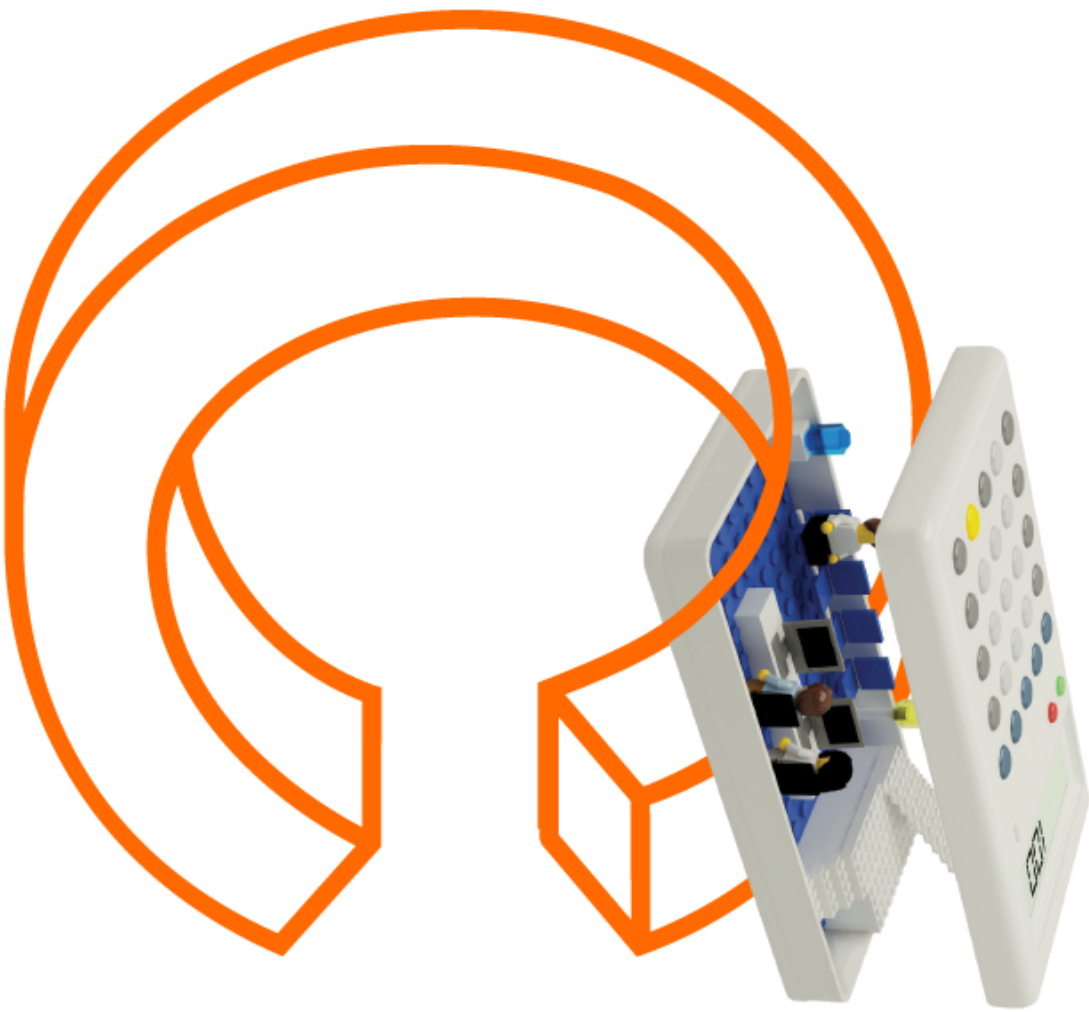


idat

Mg. Daniel Arias, **PMP™** y **Scaled Scrum Master™**

Metodología de desarrollo de software - Metodología Clásica

Semana 4



idat

Semana 4: Metodología de desarrollo de software - Metodología Clásica

Logro de aprendizaje

- Cascada
- Método Evolutivo
- Prototipos
- Desarrollo Basado en componentes
- Espiral
- Incremental

Actividades

Actividad 04:

- Elabora y presenta un Cuadro Comparativo de cada metodología Clásica

Aprendemos:

¿Qué aprendimos la clase pasada?

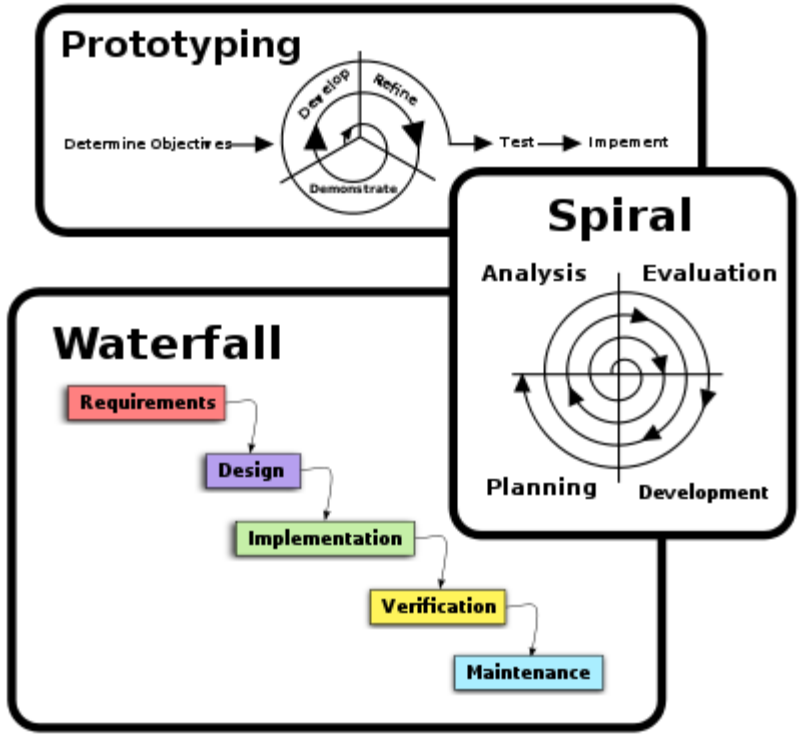
- Framework Scrum
- XP
- Kanban
- Scrumban
- Lean

http://www.menti.com

Aprendemos: Metodología de desarrollo de software - Metodología Clásica

Metodología Clásica

Las metodologías clásicas imponen una disciplina de trabajo sobre el proceso de desarrollo del software, para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar.



Aprendemos: Metodología de desarrollo de software - Metodología Clásica

Metodología Clásica

Los Objetivos de las Metodologías de Desarrollo de Sistemas de Información son:

- Definir actividades a llevarse a cabo en un Proyecto de S.I.
- Unificar criterios en la organización para el desarrollo de S.I.
- Proporcionar puntos de control y revisión
- Asegurar la uniformidad y calidad tanto del desarrollo como del sistema en sí
- Satisfacer las necesidades de los usuarios del sistema
- Conseguir un mayor nivel de rendimiento y eficiencia del personal asignado al desarrollo
- Ajustarse a los plazos y costos previstos en la planificación
- Generar de forma adecuada la documentación asociada a los sistemas
- Facilitar el mantenimiento posterior de los sistemas

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

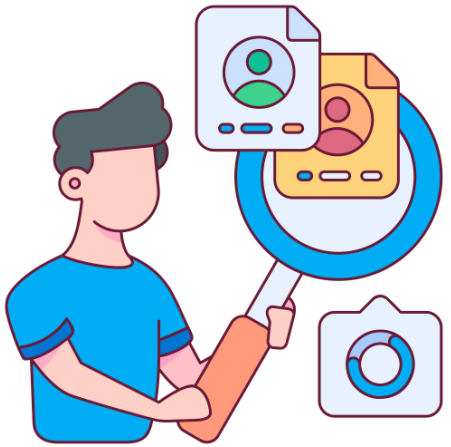
Cascada

El modelo en cascada es un proceso de desarrollo secuencial, en el que el desarrollo de software se concibe como un conjunto de etapas que se ejecutan una tras otra. Se le denomina así por las posiciones que ocupan las diferentes fases que componen el proyecto, colocadas una encima de otra, y siguiendo un flujo de ejecución de arriba hacia abajo, como una cascada

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Fases de la metodología en cascada

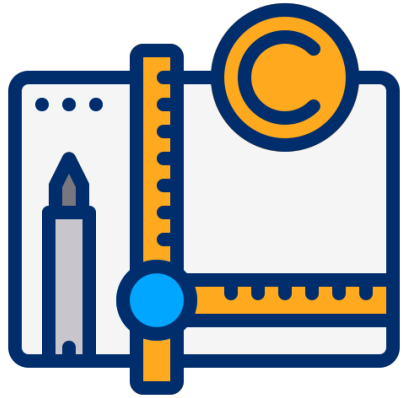
Requisitos	En esta fase se hace un análisis de las necesidades del cliente para determinar las características del software a desarrollar, y se especifica todo lo que debe hacer el sistema sin entrar en detalles técnicos. Hay que ser especialmente cuidadoso en esta primera fase, ya que en este modelo no se pueden añadir nuevos requisitos en mitad del proceso de desarrollo. Disponer de una especificación de los requisitos permite estimar de forma rigurosa las necesidades del software antes de su diseño. Además, permite tener una base a partir de la cual estimar el coste del producto, los riesgos y los plazos.



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Fases de la metodología en cascada

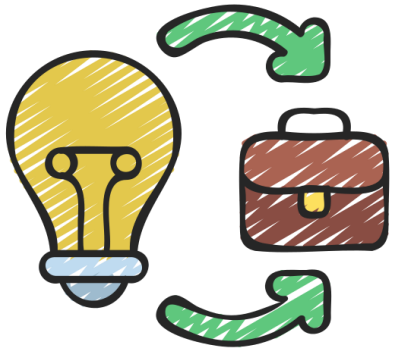
Diseño	Se describe la estructura interna del software, y las relaciones entre las entidades que lo componen. Descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD (Documento de Diseño del Software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras. Es conveniente distinguir entre diseño de alto nivel o arquitectónico y diseño detallado. El primero de ellos tiene como objetivo definir la estructura de la solución (una vez que la fase de análisis ha descrito el problema) identificando grandes módulos (conjuntos de funciones que van a estar asociadas) y sus relaciones. Con ello se define la arquitectura de la solución elegida. El segundo define los algoritmos empleados y la organización del código para comenzar la implementación



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Fases de la metodología en cascada

Implementación	En esta fase se programan los requisitos especificados haciendo uso de las estructuras de datos diseñadas en la fase anterior. La programación es el proceso que lleva de la formulación de un problema de computación, a un programa que se ejecute produciendo los pasos necesarios para resolver dicho problema. Al programar, tenemos que realizar actividades como el análisis de las condiciones, la creación de algoritmos, y la implementación de éstos en un lenguaje de programación específico. Un algoritmo es un conjunto de instrucciones o reglas bien definidas y ordenadas que permiten llevar a cabo una actividad mediante pasos sucesivos.



Fases de la metodología en cascada

Verificación	Como su propio nombre indica, una vez se termina la fase de implementación se verifica que todos los componentes del sistema funcionen correctamente y cumplen con los requisitos. El objetivo de las pruebas es el de obtener información de la calidad del software, y sirven para: encontrar defectos o bugs, aumentar la calidad del software, refinar el código previamente escrito sin miedo a romperlo o introducir nuevos bugs, etc.

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

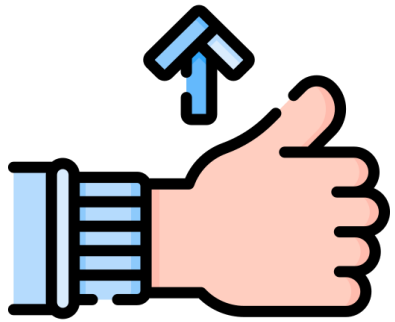
Fases de la metodología en cascada

Instalación y mantenimiento	Una vez se han desarrollado todas las funcionalidades del software y se ha comprobado que funcionan correctamente, se inicia la fase de instalación y mantenimiento. Se instala la aplicación en el sistema y se comprueba que funcione correctamente en el entorno en que se va a utilizar. A partir de ahora hay que asegurarse de que el software funcione y hay que destinar recursos a mantenerlo. El mantenimiento del software consiste en la modificación del producto después de haber sido entregado al cliente, ya sea para corregir errores o para mejorar el rendimiento o las características.

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Cascada

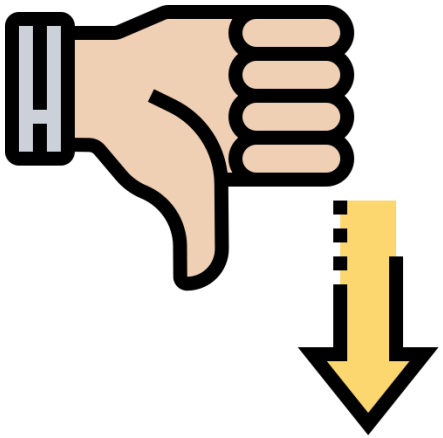
Ventajas	• El tiempo que se pasa en diseñar el producto en las primeras fases del proceso puede evitar problemas que serían más costosos cuando el proyecto ya estuviese en fase de desarrollo. • La documentación es muy exhaustiva y si se une al equipo un nuevo desarrollador, podrá comprender el proyecto leyendo la documentación. • Al ser un proyecto muy estructurado, con fases bien definidas, es fácil entender el proyecto. • Ideal para proyectos estables, donde los requisitos son claros y no van a cambiar a lo largo del proceso de desarrollo.



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Cascada

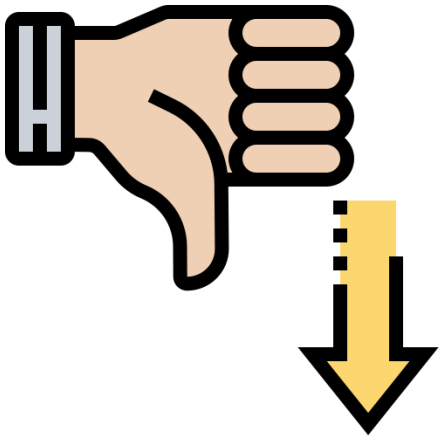
Desventajas	• En muchas ocasiones, los clientes no saben bien los requisitos que necesitarán antes de ver una primera versión del software en funcionamiento. Entonces, cambiarán muchos requisitos y añadirán otros nuevos, lo que supondrá volver a realizar fases ya superadas y provocará un incremento del coste. • No se va mostrando al cliente el producto a medida que se va desarrollando, si no que se ve el resultado una vez ha terminado todo el proceso. Esto cual provoca inseguridad por parte del cliente que quiere ir viendo los avances en el producto



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Cascada

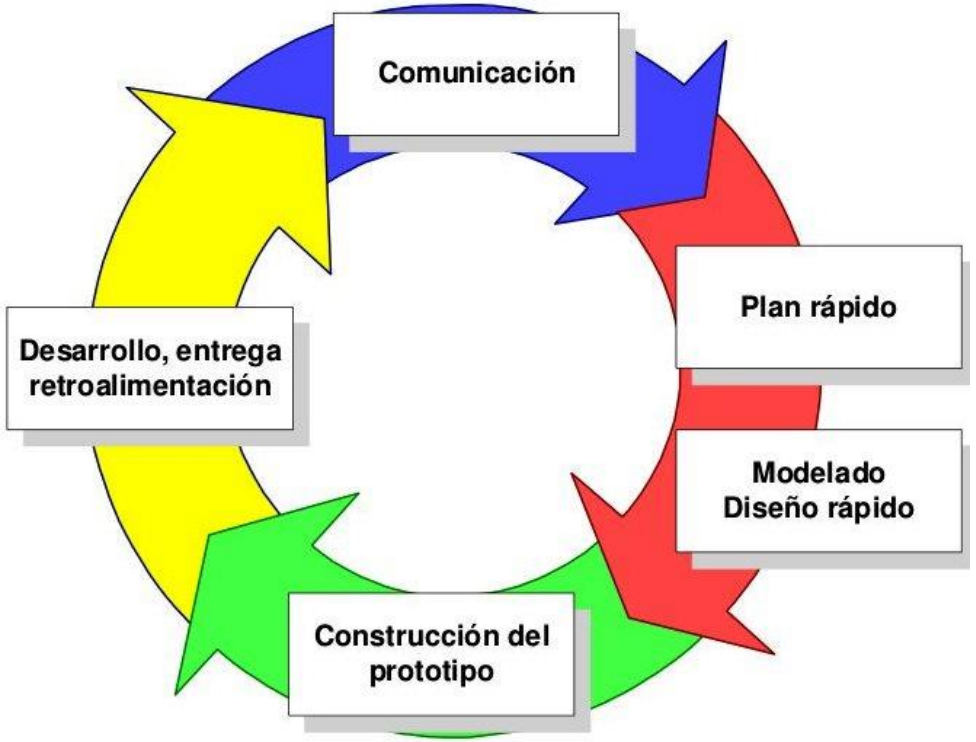
Desventajas	• Los diseñadores pueden no tener en cuenta todas las dificultades que se encontrarán cuando estén diseñando un software, lo que conllevará rediseñar el proyecto para solventar el problema. • Para proyectos a largo plazo, este modelo puede suponer un problema al cambiar las necesidades del usuario a lo largo del tiempo. Si, por ejemplo, tenemos un proyecto que va a durar 5 años, es muy probable que los requisitos necesiten adaptarse a los gustos y novedades del mercado.



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Método Evolutivo

Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Un sistema inicial se desarrolla rápidamente a partir de especificaciones abstractas. Éste se refina basándose en las peticiones del cliente para producir un sistema que satisfaga sus necesidades.



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Método Evolutivo

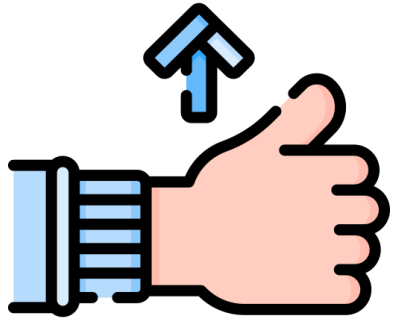
La idea detrás de este modelo es el desarrollo de una implantación del sistema inicial, exponerla a los comentarios del usuario, refinarla en N versiones hasta que se desarrolle el sistema adecuado. Una ventaja de este modelo es que se obtiene una rápida realimentación del usuario, ya que las actividades de especificación, desarrollo y pruebas se ejecutan en cada iteración.

El desarrollo evolutivo consta del desarrollo de una versión inicial que luego de exponerse se va refinando de acuerdo de los comentarios o nuevos requerimientos por parte del cliente o del usuario final. Las fases de especificación, desarrollo y validación se entrelazan en vez de separarse.

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Método Evolutivo

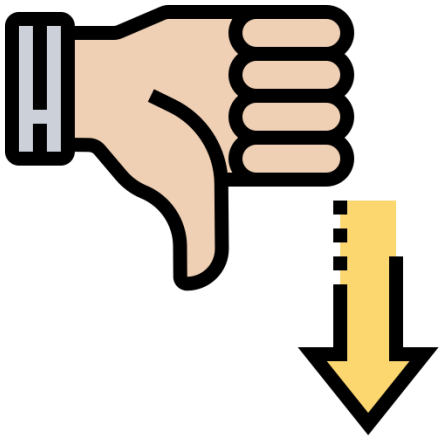
Ventajas	• La especificación puede desarrollarse de forma creciente. • Los usuarios y desarrolladores logran un mejor entendimiento del sistema. Esto se refleja en una mejora de la calidad del software. • Es más efectivo que el modelo de cascada, ya que cumple con las necesidades inmediatas del cliente. • Hay que implicar a los usuarios.



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Método Evolutivo

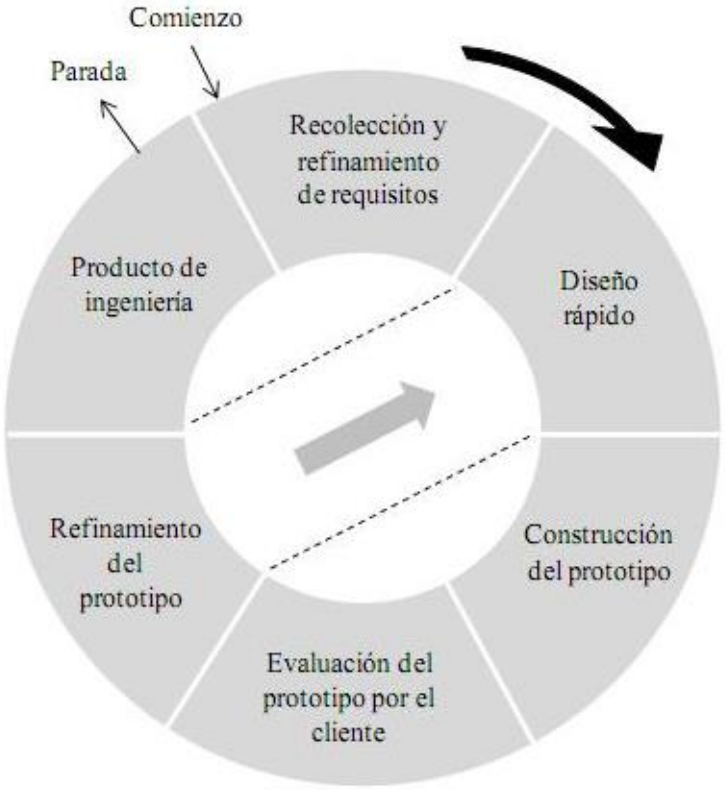
Desventajas	• Es difícil predecir el coste y duración de un proyecto, por lo que es conveniente limitar el número de fases, recursos y plazos si es un proyecto con principio y fin. • Puede resultar costoso si hay que reiniciar el desarrollo. • Hay que mantener muy bien la documentación del proyecto para facilitar el control de versiones y su mantenimiento • Para el rápido desarrollo se necesitan herramientas que pueden ser incompatibles con otras o que poca gente sabe utilizar.



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Prototipo

La creación de prototipos consiste en construir rápida y económicamente un sistema experimental para que lo evalúen los usuarios finales. Interactuando con el prototipo, los usuarios pueden darse una mejor idea de sus requerimientos de información. El prototipo avalado por los usuarios puede servir de plantilla para crear el sistema definitivo.



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Prototipo

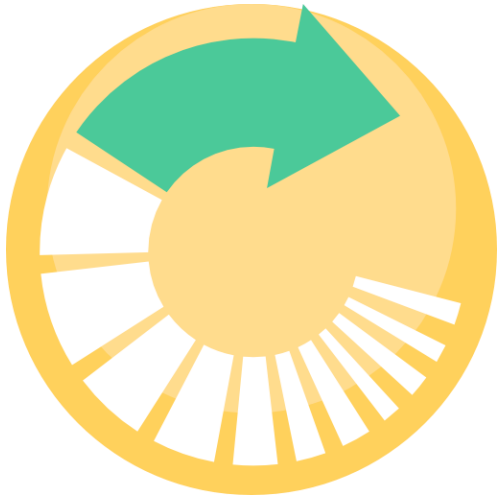
El prototipo es una versión funcional de un sistema de información o de parte de éste, pero su propósito es el de servir de modelo preliminar. Una vez en operación, el prototipo se refinará más aún hasta que cumpla con precisión los requerimientos de los usuarios. Una vez finalizado el diseño, el prototipo se puede convertir en un sistema en producción refinado. Según esto un prototipo puede tener alguna de las tres formas siguientes:

- Un prototipo, en papel o ejecutable en ordenador, que describa la interacción hombre-máquina y los listados del sistema.
- Un prototipo que implemente algún(os) subconjunto(s) de la función requerida, y que sirva para evaluar el rendimiento de un algoritmo o las necesidades de capacidad de almacenamiento y velocidad de cálculo del sistema final.
- Un programa que realice en todo o en parte la función deseada pero que tenga características (rendimiento, consideración.

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Prototipo - Etapas

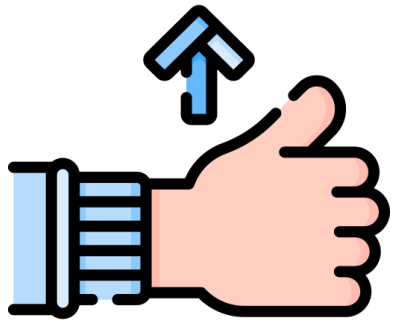
- Recolección y refinamiento de requisitos
- Modelado, diseño rápido
- Construcción del Prototipo
- Desarrollo, evaluación del prototipo por el cliente
- Refinamiento del prototipo
- Producto de Ingeniería



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Prototipo

Ventajas	• No modifica el flujo del ciclo de vida • Reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios • Reduce costo y aumenta la probabilidad de éxito • Exige disponer de las herramientas adecuadas • Este modelo es útil cuando el cliente conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida. • También ofrece un mejor enfoque cuando el responsable del desarrollo del software está inseguro de la eficacia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma que debería tomar la interacción humano-máquina.



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

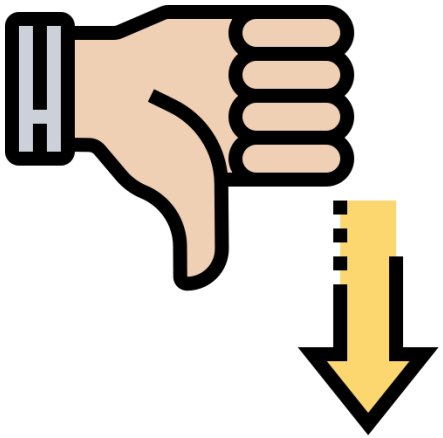
Prototipo

Para que sea efectivo	• Debe ser un sistema con el que se pueda experimentar • Debe ser comparativamente barato (menor que el 10%) • Debe desarrollarse rápidamente • Énfasis en la interfaz de usuario • Equipo de desarrollo reducido • Herramientas y lenguajes adecuadas

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Prototipo

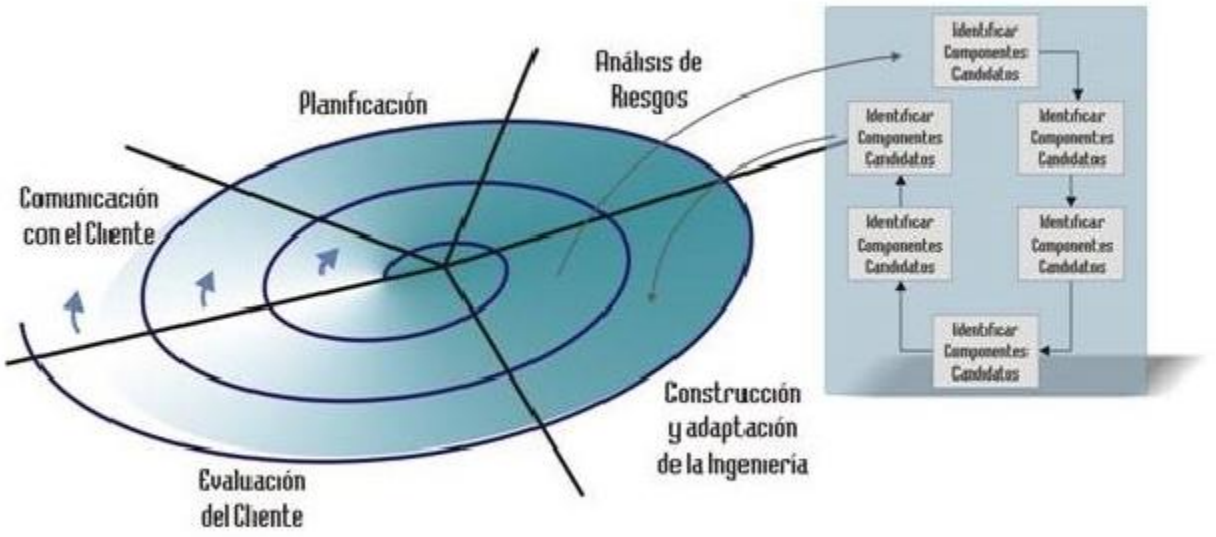
Desventajas	• Debido a que el usuario ve que el prototipo funciona piensa que este es el producto terminado y no entienden que recién se va a desarrollar el software. • El desarrollador puede caer en la tentación de ampliar el prototipo para construir el sistema final sin tener en cuenta los compromisos de calidad y mantenimiento que tiene con el cliente.



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Desarrollo Basado en componentes

La metodología de software basada en Componentes surgió a finales de los 90's como una aproximación basada en la reutilización al desarrollo de sistemas de software.



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Desarrollo Basado en componentes

El desarrollo de software basado en componentes permite reutilizar piezas de código preelaborado que permiten realizar diversas tareas, conllevando a diversos beneficios como las mejoras a la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión. Es básicamente como cuando (si es que llevas tiempo programando) tienes ya una clase o un código guardado en algún lugar y te acuerdas de el, de manera que copias y pegas el código en tu proyecto para usarlo, o lo conviertes en una clase y mandas llamar sus métodos, de esta manera podemos decir que es la metodología que recicla códigos.

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Desarrollo Basado en componentes

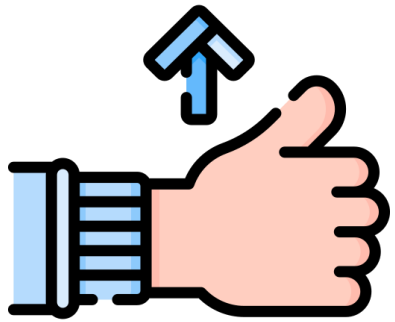
Un Componente debe ser:

- **Identificable:** Debe tener una identificación que permita acceder fácilmente a sus servicios que permita su clasificación.
- **Auto contenido:** Un componente no debe requerir de la utilización de otros para finiquitar la función para la cual fue diseñado.
- **Puede ser remplazado por otro componente:** Se puede remplazar por nuevas versiones u otro componente que lo remplace y mejore.
- **Con acceso solamente a través de su interfaz:** Debe asegurar que estas no cambiaran a lo largo de su implementación.
- **Sus servicios no varían:** Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación sí.
- **Bien Documentado:** Un componente debe estar correctamente documentado para facilitar su búsqueda si se quiere actualizar, integrar con otros, adaptarlo, etc.
- **Es genérico:** Sus servicios deben servir para varias aplicaciones.
- **Reutilizado dinámicamente:** Puede ser cargado en tiempo de ejecución en una aplicación.
- **Independiente de la plataforma:** Hardware, Software, S.O.

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

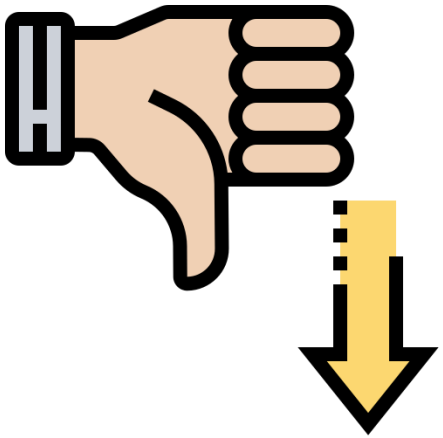
Desarrollo Basado en componentes

Ventajas	• Reutilización del software. • Simplifica las pruebas. • Simplifica el mantenimiento del sistema. • Mayor calidad. • Ciclos de desarrollo más cortos. • Mejor ROI. • Funcionalidad mejorada.



Desarrollo Basado en componentes

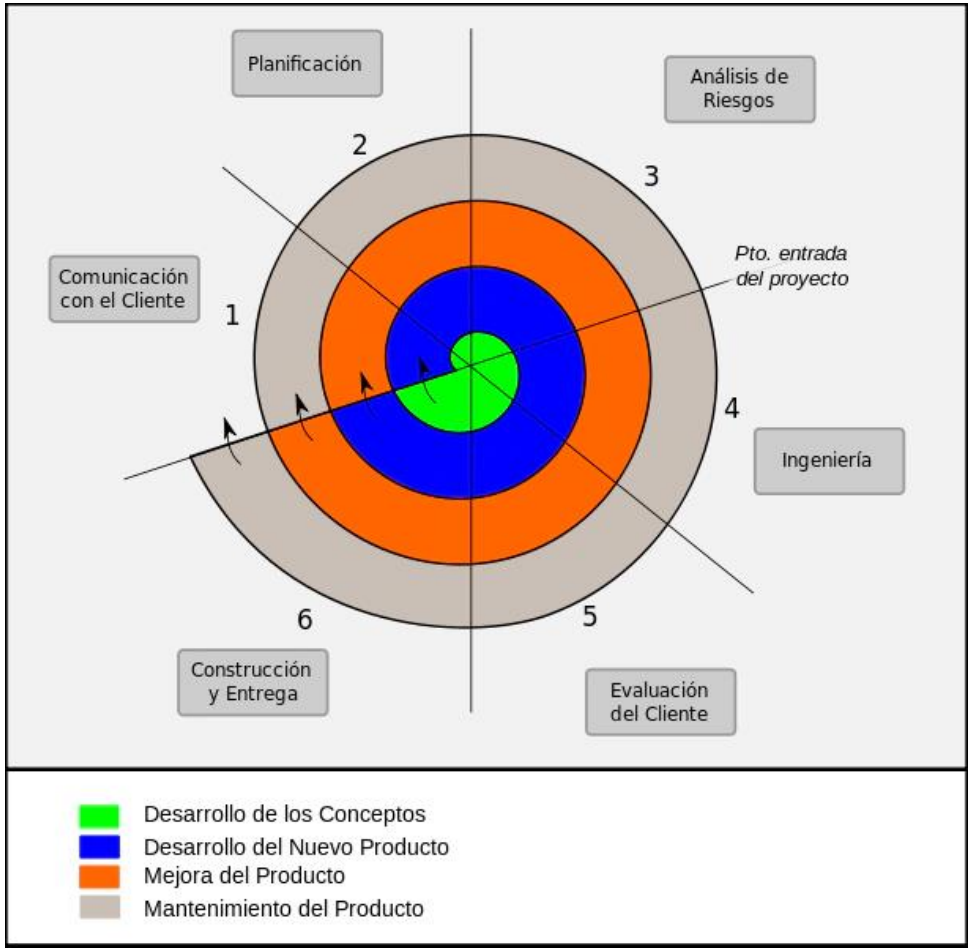
Desventajas	• Genera mucho tiempo. • Genera mucho trabajo adicional. • Confiabilidad de los componentes. • Los componentes son cajas negras de unidades de programas, y el código de los componentes puede no estar disponible para los usuarios de dichos componentes.



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Espiral

En el modelo espiral, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones la versión incremental podría ser un modelo en papel o un prototipo, durante las últimas iteraciones se producen versiones cada vez más completas del sistema diseñado.



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Espiral

Una característica clave del desarrollo en espiral es la minimización de los riesgos en el desarrollo de software, lo que podría resultar en un aumento de los costes totales, más esfuerzo y un lanzamiento retardado. Estos riesgos son contrarrestados por el enfoque incremental, haciendo primero prototipos, que luego pasan al menos una vez, por las fases de desarrollo de software. El desarrollo en espiral es genérico y puede combinarse con otros métodos de desarrollo clásicos y ágiles, por lo que también se denomina modelo o desarrollo de segundo orden.
EL modelo en espiral se divide en un número de actividades de marco de trabajo, también llamadas REGIONES DE TAREAS, Cada una de las regiones están compuestas por un conjunto de tareas del trabajo llamado CONJUNTO DE TAREAS que se adaptan a las características del proyecto que va a emprenderse en todos los casos se aplican actividades de protección.

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Ciclos en Espiral

- **Objetivo y determinación alternativa:** Los objetivos se determinan conjuntamente con el cliente. Al mismo tiempo, se discuten posibles alternativas y se especifican las condiciones marco (por ejemplo, sistemas operativos, entornos y lenguajes de programación).
- **Análisis y evaluación de riesgos:** Se identifican y evalúan los riesgos potenciales. También se evalúan las alternativas existentes. Los riesgos son registrados, evaluados y luego reducidos utilizando prototipos, simulaciones y softwares de análisis. En este ciclo, existen varios prototipos como plantillas de diseño o componentes funcionales
- **Desarrollo y prueba:** Los prototipos se amplían y se añaden funcionalidades. El código real es escrito, probado y migrado a un entorno de prueba varias veces hasta que el software pueda ser implementado en un entorno productivo.
- **Planificación del siguiente ciclo:** Se planifica al final de cada etapa. Si se producen errores, se buscan soluciones, y si una alternativa es una mejor solución, se prefiere en el siguiente ciclo.

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Espiral

Ventajas / Desventajas	• Se utiliza a menudo para proyectos más grandes que están sujetos a riesgos. Dado que estos riesgos tienen un impacto monetario directo, el control de los presupuestos de los clientes y de las empresas promotoras es fundamental. Se utiliza especialmente en los nuevos entornos técnicos, ya que éstos suponen un riesgo. • Los conflictos entre los requisitos de un software y su diseño se evitan eficazmente mediante el enfoque cíclico, ya que los requisitos pueden comprobarse constantemente y, si es necesario, modificarse.

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

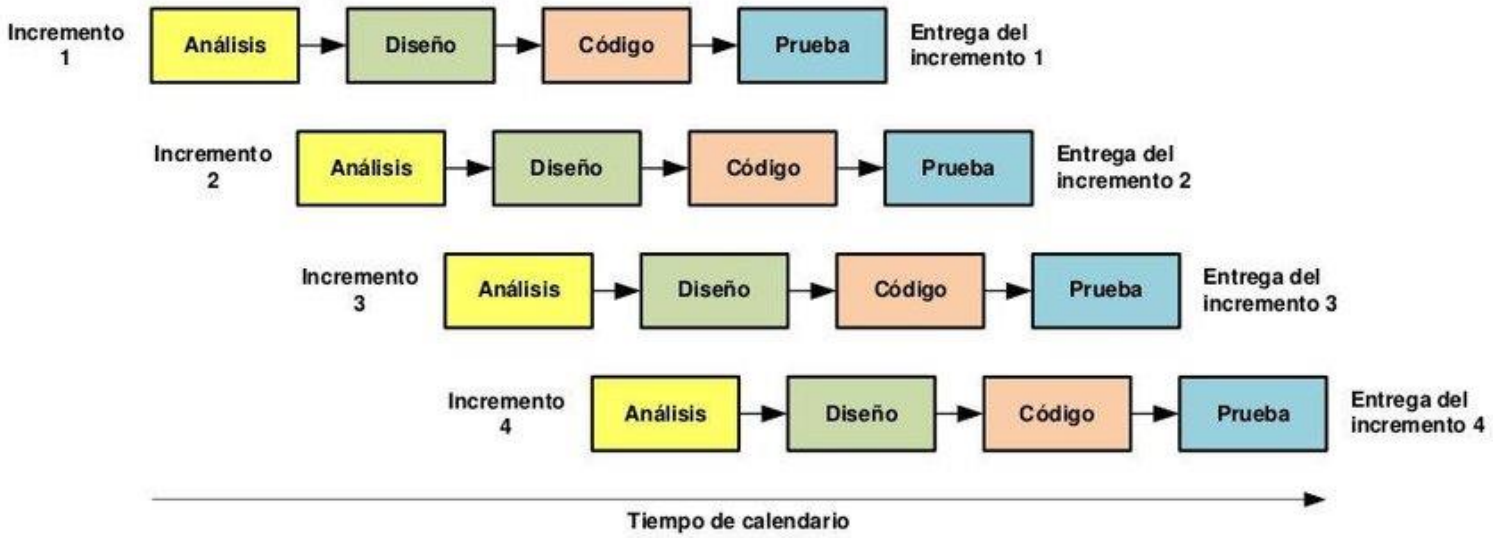
Espiral

Ventajas / Desventajas	• Se puede obtener feedback de los usuarios, desarrolladores y clientes en las primeras fases del proyecto. Sin embargo, esta estructura también requiere una gestión que tenga en cuenta los ciclos del producto y pueda responder rápidamente a los riesgos. El control de tales proyectos es, por lo tanto, relativamente complejo y también requiere una buena documentación para que se registren todos los cambios. • Aunque el software se prueba bajo varios aspectos durante el ciclo de desarrollo y prueba (unidad, prueba de aceptación e integración), a menudo sucede que los prototipos se transfieren al sistema de producción. Por lo tanto, existe el riesgo de que se introduzcan otros errores e incoherencias conceptuales en el producto final posterior. • En los lugares donde se toman decisiones sobre los ciclos siguientes, existe el riesgo de que se formen bucles y el proyecto tarde más tiempo si se toman decisiones equivocadas. Por esta razón, las alternativas y su evaluación son importantes.

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

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. En cada incremento, el producto debe mostrar una evolución con respecto a la fecha anterior; nunca puede ser igual.



Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

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.

La principal diferencia del modelo incremental con los modelos tradicionales es que las tareas están divididas en iteraciones, es decir, pequeños lapsos en los cuales se trabaja para conseguir objetivos específicos. Con los modelos tradicionales no pasaba esto; era necesario esperar hasta el final del proceso.

Aprendemos: Metodología de desarrollo de software – Tipos de Metodologías Clásicas

Incremental - Características

- Los incrementos son pequeños.
- Permite una fácil administración de las tareas en cada iteración.
- La inversión se materializa a corto plazo.
- Es un modelo propicio a cambios o modificaciones.
- Se adapta a las necesidades que surjan.

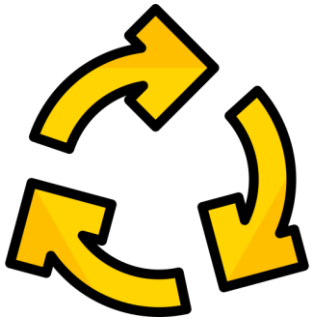
Para que esto sea posible, se debe tener en cuenta que las iteraciones no pueden ser demasiado rígidas y que no existan tareas simultáneas. El modelo incremental exige un encadenamiento progresivo de cada tarea. Scrum y Kanban son las herramientas más conocidas que emplean este modelo de gestión.

Incremental - Fases

- **Requerimientos:** son los objetivos centrales y específicos que persigue el proyecto.
- **Definición de las tareas y las iteraciones:** teniendo en cuenta lo que se busca, el siguiente paso es hacer una lista de tareas y agruparlas en las iteraciones que tendrá el proyecto. Esta agrupación no puede ser aleatoria. Cada una debe perseguir objetivos específicos que la definan como tal.
- **Diseño de los incrementos:** establecidas las iteraciones, es preciso definir cuál será la evolución del producto en cada una de ellas. Cada iteración debe superar a la que le ha precedido. Esto es lo que se denomina incremento.
- **Desarrollo del incremento:** posteriormente se realizan las tareas previstas y se desarrollan los incrementos establecidos en la etapa anterior.

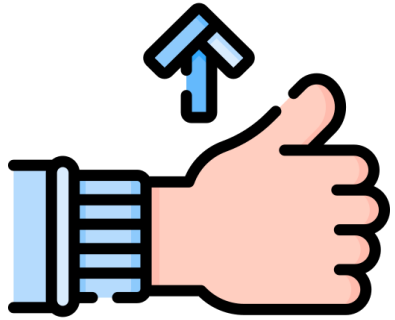
Incremental - Fases

- **Validación de incrementos:** al término de cada iteración, los responsables de la gestión del proyecto deben dar por buenos los incrementos que cada una de ellas ha arrojado. Si no son los esperados o si ha habido algún retroceso, es necesario volver la vista atrás y buscar las causas de ello.
- **Integración de incrementos:** una vez son validados, los incrementos dan forma a lo que se denomina línea incremental o evolución del proyecto en su conjunto. Cada incremento ha contribuido al resultado final.
- **Entrega del producto:** cuando el producto en su conjunto ha sido validado y se confirma su correspondencia con los objetivos iniciales, se procede a su entrega final.



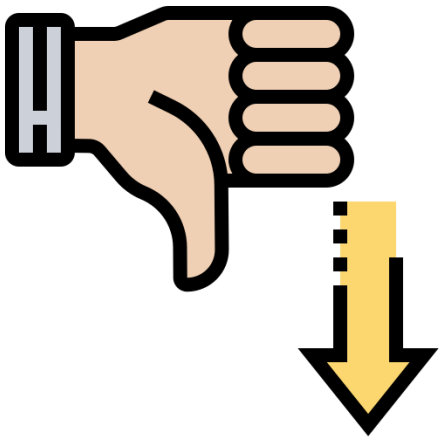
Incremental

Ventajas	• Mediante este modelo se genera software operativo de forma rápida y en etapas tempranas del ciclo de vida del software. • Es un modelo más flexible, por lo que se reduce el coste en el cambio de alcance y requisitos. • Es más fácil probar y depurar en una iteración más pequeña. • Es más fácil gestionar riesgos. • Cada iteración es un hito gestionado fácilmente.



Incremental

Desventajas	• Cada fase de una iteración es rígida y no se superponen con otras. • Pueden surgir problemas referidos a la arquitectura del sistema porque no todos los requisitos se han reunido, ya que se supone que todos ellos se han definido al inicio.



Verificamos lo aprendido:

Kahoot!

http://www.kahoot.it/

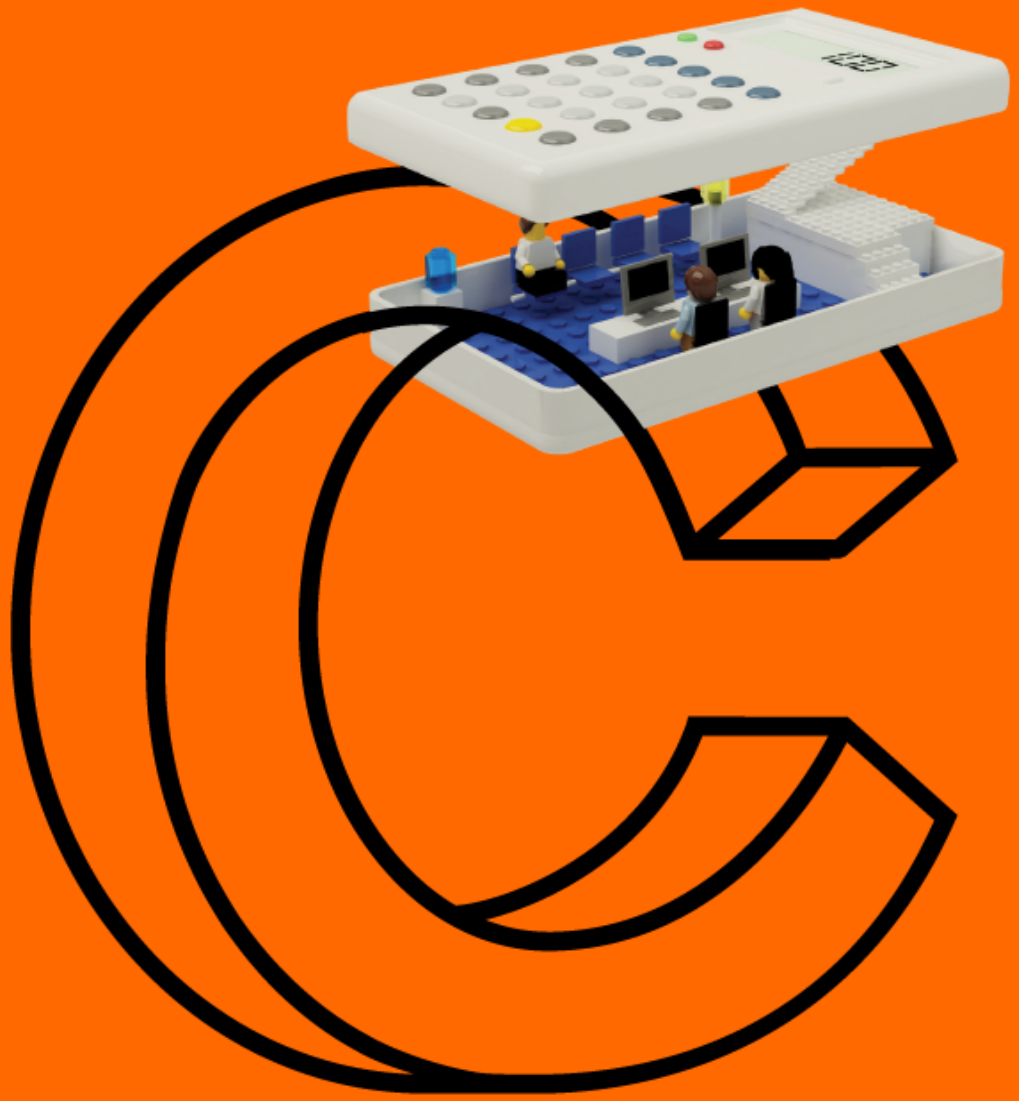
Verificamos lo aprendido:

- Absolución de Preguntas.
- Conclusiones y Recomendaciones.

Aplicamos lo aprendido: Asignación para la clase siguiente

- Comentar en el Foro

Gracias



idat