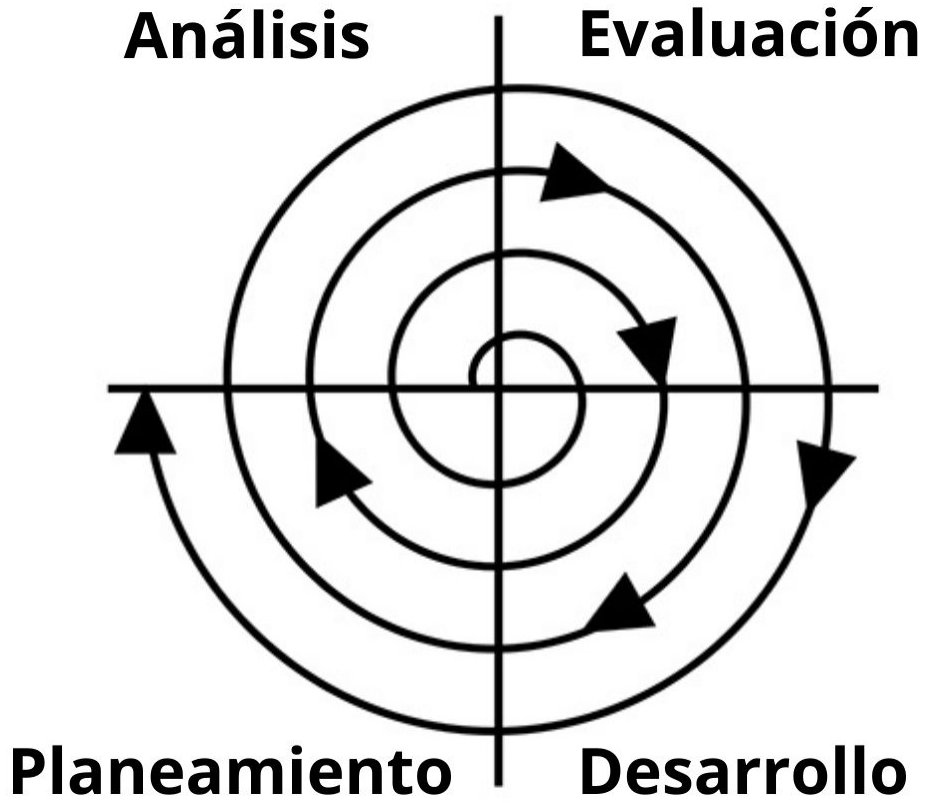
# Ciclos de Vida del Software

Modelo espiral (Fila 1)




## Información general

Se trata de una combinación de los modelos cascada y el de iteraciones.
El proceso pasa por distintas etapas, comienza con un análisis de objetivos, seguido de una evaluación de riesgos, después se realizará el desarrollo y pruebas del software, y por último la planificación de la siguiente fase. Estas etapas se repiten hasta que se consigue finalizar el software .
Dentro de cada etapa, tendremos una serie de fases que transcurren desde la planificación, pasando por el análisis de riesgos, el desarrollo y finalizando en la evaluación de lo realizado. Se incorpora también una fase de enlace entre etapas, para facilitar la transición entre las mismas.
En definitiva, el equipo de desarrollo en este modelo de desarrollo en espiral comienza con un pequeño conjunto de requisitos y pasa por cada fase de desarrollo para ese conjunto de requisitos. El equipo de desarrollo agrega la funcionalidad para el requerimiento adicional en espirales cada vez mayores, hasta que la aplicación está lista para la fase de producción.

# Cómo funciona el modelo en espiral

El modelo de desarrollo en espiral es una combinación entre el modelo de cascada y el de interacciones. El modelo en espiral describe el ciclo de vida de un software por medio de espirales, que se van repitiendo hasta llegar a hacer el producto terminado.

El modelo de desarrollo en espiral se caracteriza por los siguientes ciclos (también cuadrantes):

**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:** El 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.

El objetivo del ciclo es producir un producto en continua mejora. El software o la aplicación se perfecciona constantemente. El modelo en espiral es incremental, pero no necesariamente repetitivo. Las repeticiones ocurren sólo cuando los riesgos, errores o conflictos amenazan el proyecto.

# BENEFICIOS Y DESVENTAJAS

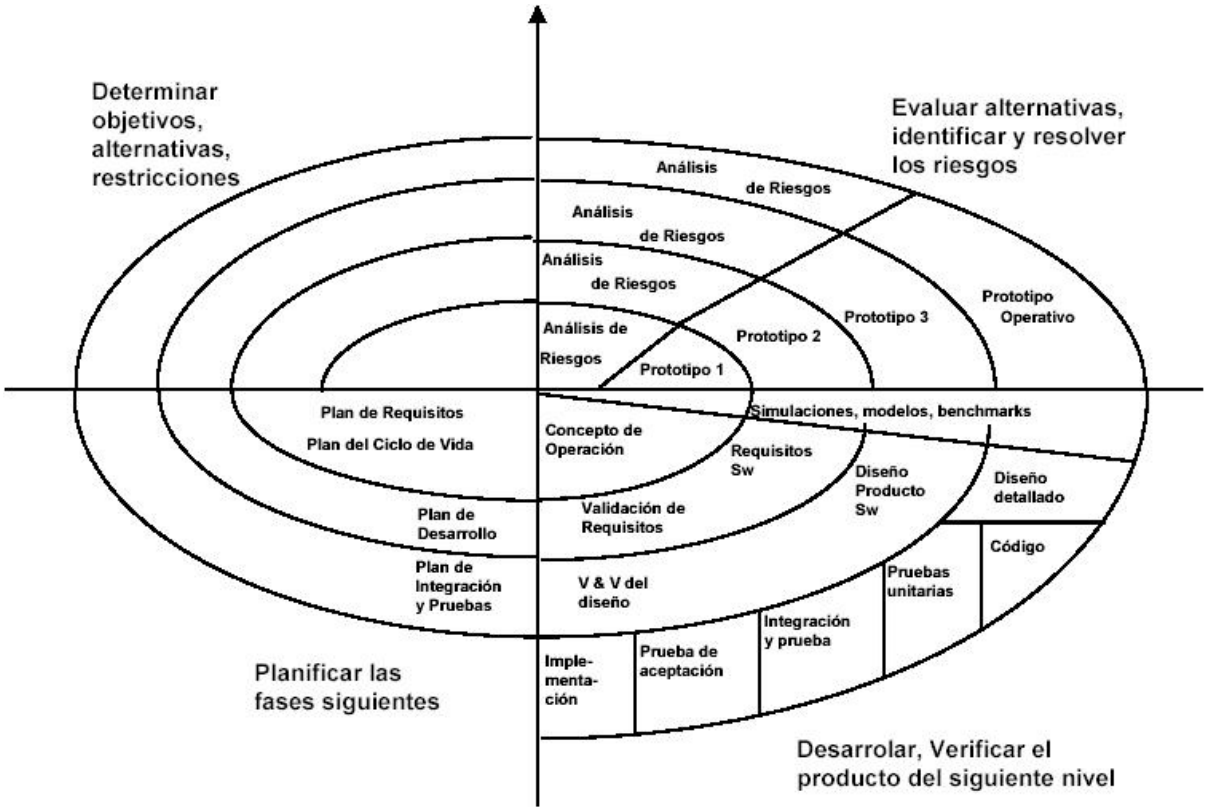
- Se utiliza a menudo para proyectos más grandes que tienden a tener más riesgos. Dado que estos riesgos tienen un impacto monetario directo, el control de los presupuestos de los clientes y de las empresas promotoras es fundamental. El modelo en espiral se utiliza especialmente en los nuevos entornos técnicos, ya que suponen un riesgo.

- Los conflictos entre los requisitos de un software y su diseño se evitan eficazmente mediante el enfoque en espiral, ya que los requisitos pueden comprobarse constantemente y, si es necesario, modificarse.

- 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 los proyectos es 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, 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.

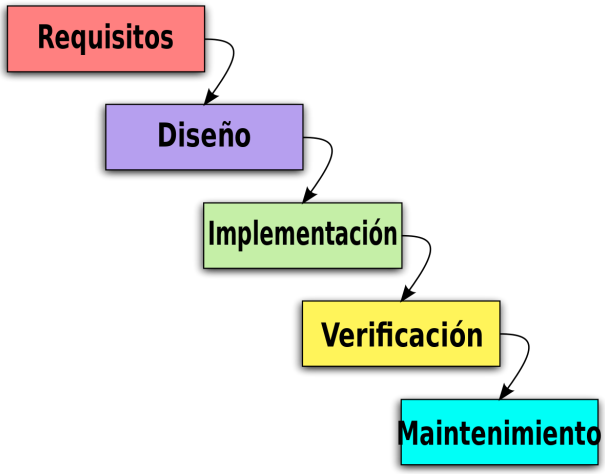
- En los lugares donde se toman decisiones sobre los ciclos siguientes, existe el riesgo de que se formen bucles y el proyecto tarde más tiempo si se toman decisiones equivocadas. Por eso, las alternativas y sus evaluaciones son importantes.




## Relevancia para la programación

A diferencia de un modelo secuencial (por ejemplo, el modelo en cascada) dispuesto en fases sucesivas, el modelo en espiral esboza el ciclo de vida de un software por medio de espirales que hay que recorrer. Por lo tanto, este enfoque se parece más a la creación de prototipos que a los enfoques clásicos. El modelo en espiral se supone que evita las desventajas de otros modelos y enfatiza las ventajas. Al centrarse en la minimización del riesgo, este modelo tiene un componente financiero que puede ser relevante para los responsables de la toma de decisiones. A través de discusiones con los clientes, análisis y estudios de viabilidad, se pueden implementar proyectos a gran escala y monitorear su impacto económico. Hasta qué punto los métodos ágiles como scrum, programación extrema o DevOps son mejores opciones dependen de muchos factores diferentes como el alcance del proyecto, el presupuesto o el nivel requerido de soporte y mantenimiento. Cuanto más flexibilidad se requiera, más métodos ágiles se considerarán.

# Modelo en Cascada (Fila 3)




También llamado modelo de desarrollo clásico, es un método de gestión de proyectos de software, en el que se divide en distintas fases secuenciales, donde puede pasarse a la siguiente fase sólo cuando se haya completado la anterior.

La clave del modelo en cascada es que no hay posibilidad de cambios o errores, por lo que lo fundamental es la planificación. La calidad de trabajo de inicio define en mayor medida el resultado final.

La mayor responsabilidad del jefe de proyecto es elaborar bien todos los requisitos y prever todas las preguntas pertinentes. Los requisitos deben ser lo más completos posibles porque el equipo trabaja con la investigación y el diseño en las etapas iniciales.

**¿Cuándo se aplica el método de cascada?**

- Cuando hay una visión clara de lo que debe ser el software final.
- Cuando los clientes no tienen posibilidad de cambiar el alcance del proyecto una vez que ha comenzado.
- Cuando el concepto y la definición son las claves del éxito, pero **NO** la velocidad.
- Cuando no hay requisitos ambiguos.

**Ventajas:**

- El modelo es simple y fácil de usar.
- Como el modelo es bastante rígido, es fácil de administrar.
- El proceso es bastante predecible.
- Las fases no se superponen, se ejecutan y se completan una a la vez.
- Los modelos de desarrollo de software en cascada son buenos para proyectos que contienen requisitos claros.
- Si la rotación de empleados en la empresa es bastante frecuente, impactará mínimamente el software.

**Desventajas:**

- Si se encuentra un error o necesita cambiar algo, el proyecto debe iniciarse desde el principio con un nuevo código.
- Cuando el software está en la etapa de prueba, no es fácil volver atrás.
- No puede resolver algunos problemas esenciales utilizándolo para proyectos complejos y orientados a objetos. Tampoco es una buena idea usarlo para proyectos muy largos.
- El método no es apropiado para los proyectos en los que desde el inicio hay muchas probabilidades de que los requisitos cambien.
- Como se basa en requisitos documentados, es posible que los clientes no vean lo que se entregará hasta que esté casi terminado, por lo que si hay algo con lo que están descontentos, puede ser difícil cambiarlo en ese momento.

**1.Requisitos**

En los requisitos de un modelo de desarrollo en cascada o desarrollo clásico es una fase donde se realiza una evaluación de los costes , estudio de la rentabilidad y la factibilidad del proyecto de sofware.Se necesitaria tambien un plan y una estimación del coste del proyecto, así como decirselo al propio cliente si es necesario o lo pide.

Después de hacer todo lo anterior y comprobar que se está listo para empezar se necesitaría  hacer una definición detallada de los requisitos(Análisis de situación de salida y un concepto).
Hacer un análisis de la situación de salida se encarga de descubrir los problemas que puede haber cuando se está realizando el software mientras el concepto se encarga de definir las funciones y características que debe hacer el producto para cumplir con las expectativas.Es decir, hacer una funcion detallada de los requisitos es hacer una descripción de los requisitos del proyecto que  se deben cumplir y un plan para continuar el proyecto.

Por último se debe hacer un análisis de los requisitos,que sirve para que cuando haya problemas  difíciles o complejos se dividan en partes pequeñas para solucionarlo con estrategias de resolución.

**2. Diseño.**

La fase de diseño sirve para formular una solución específica en base a las exigencias, tareas y estrategias definidas en la fase anterior. En esta fase, los desarrolladores de software se encargan de diseñar la arquitectura de software, así como un plan de diseño detallado del mismo, centrándose en componentes concretos. Todo esto da como resultado un borrador con el plan de diseño del software junto con planes de prueba para los diferentes componentes. Durante esta fase se debe:

- Definir la organización de la estructura y la de todos los elementos que necesitas para el desarrollo de tu software.
- Describir cómo se relacionan cada uno de los elementos entre sí para que funcionen de manera correcta, teniendo siempre en cuenta el diseño de la interfaz.

**3.Implementación**

En la fase anterior (fase de diseño), concebimos la arquitectura del software, que vamos a implementar/desarrollar durante la fase de implementación. En esta fase, se programará el software final, las pruebas unitarias y la búsqueda de errores.
En la fase de implementación se traduce el software a uno o varios lenguajes de programación, se divide en diferentes partes o componentes, a los que posteriormente se les someterá a pruebas unitarias, en la búsqueda de errores para verificar su buen funcionamiento e implementarlo poco a poco al software o programa final.
La fase de implementación da como resultado un software final operativo, una versión alfa/beta, por así decirlo.

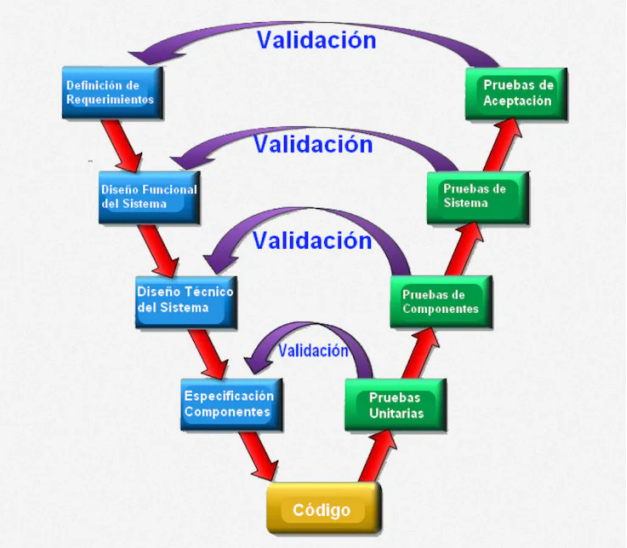
**4.Verificación**
En la verificación se realiza un prueba de integración de software. Los productos de software se envían a ciertos usuarios para que realicen la versión beta. Esto permite determinar si el software cumple con las exigencias de los usuarios beta. Si el producto cumple esas exigencias está listo para ser lanzado.

**5. Mantenimiento.**

En la última fase, se deben analizar los resultados obtenidos en el paso anterior y, de ser necesario, realizar los ajustes pertinentes. Después de esto, se entrega el producto final al cliente si funciona tal y como debería. Es probable que se deba regresar a esta fase del modelo en cascada más de una vez cada cierto tiempo para comprobar que se adapta a los cambios de su entorno.

# Modelo en V (Fila 4)

El **ciclo en V**, también llamado método en V o modelo en V, se define como un modelo de gestión de proyectos.
El modelo en V es un proceso que **representa la secuencia de pasos en el desarrollo del ciclo de vida de un proyecto**. Contribuye al cumplimiento integral de las funcionalidades, especificaciones y diseño del software, en conformidad con lo requerido por el cliente.
**Inspirado en el modelo en cascada** (Waterfall model en inglés), representa todo el ciclo de vida de un proyecto y se basa en el mismo principio de gestión secuencial y lineal.
Consta de:
Una fase descendente donde se señalan las **necesidades del proyecto**;
una fase ascendente donde se especifican las **verificaciones de las necesidades**.
Este es el aspecto del método V:




El **Método-V** fue desarrollado para regular el proceso de desarrollo de software por la **Administración Federal Alemana**. Describe las actividades y los resultados que se producen durante el desarrollo del software.

Proporciona una **guía para la planificación y realización de proyectos**. Los siguientes objetivos están destinados a ser alcanzados durante la ejecución del proyecto:

- Minimización de los riesgos del proyecto
- Mejora y Garantía de Calidad
- Reducción de los gastos totales durante todo el proyecto y sistema de Ciclo de Vida
- Mejora de la comunicación entre todos los inversionistas

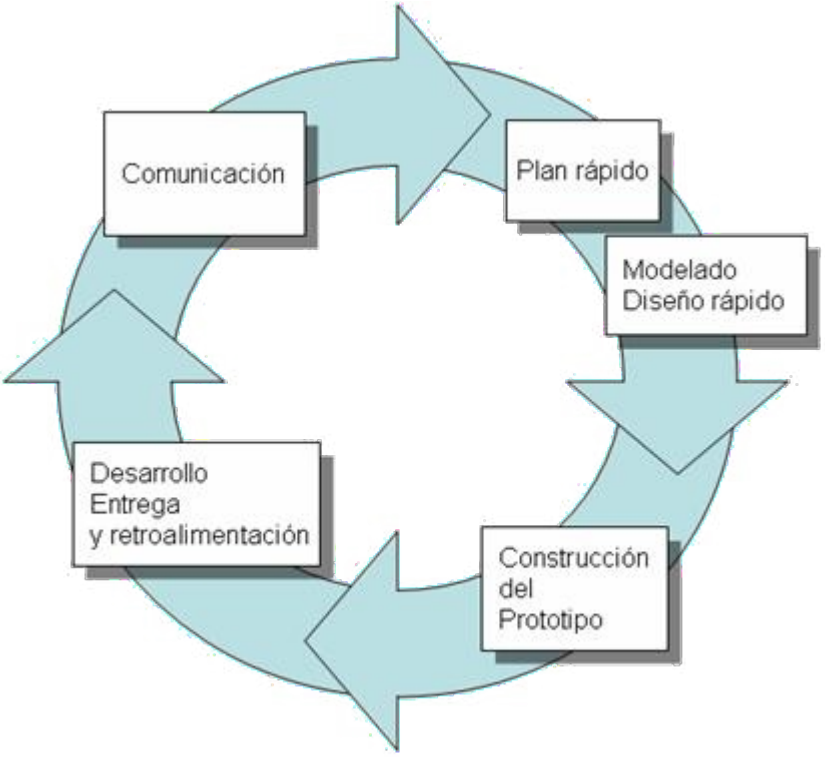
Hoy en día, son muchas las estructuras que utilizan el Método-V para la gestión de sus proyectos:

- Sectores militares.
- Sectores públicos.
- Sectores de desarrollo.
- Sectores de investigación.

# Metodología Prototipado

A menudo los requisitos no están especificados claramente:

- por no existir experiencia previa.
- por omisión o falta de concreción del usuario/cliente.




Proceso:

- Se crea un prototipo durante la fase de análisis y es probado por el usuario/cliente para refinar los requisitos del software a desarrollar.
- Se repite el paso anterior las veces necesarias.

Tipos de prototipos:

- Prototipos rápidos
    - El prototipo puede estar desarrollado usando otro lenguaje y/o herramientas.
    - Finalmente, el prototipo se desecha.
- Prototipos evolutivos
    - El prototipo está diseñado en el mismo lenguaje y herramientas del proyecto.
    - El prototipo se usa como base para desarrollar el proyecto.

# *Metodología Agile Scrum (Fila 2)*

## ***El Proceso***

En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija (iteraciones que normalmente son de 2 semanas). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final para que tenga las condiciones necesarias para ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

## ***Fases de la metodología Scrum***

La metodología Scrum pasa por diferentes fases que hacen posible que se lleve a cabo con éxito.

### 1. Planificación: Product Backlog

El Product Backlog es la fase en la que se establecen las tareas prioritarias y donde se obtiene información breve y detallada sobre el proyecto que se va a desarrollar.

El Product Backlog es necesario para poder arrancar con el primer sprint, tiene permitido cambiar y crecer tantas veces como sea necesario en función del aprendizaje adquirido en el desarrollo del producto.

### 2. Ejecución: Sprint

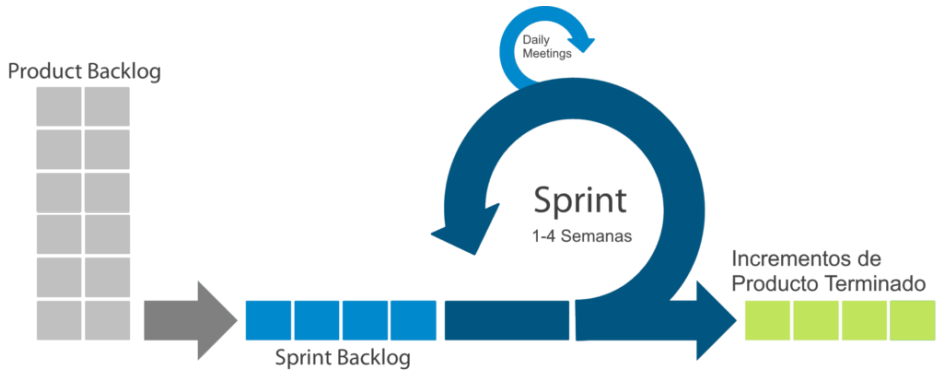
Dentro del método Scrum, **el Sprint es el corazón**, un intervalo de tiempo que como máximo tiene una duración de un mes y en donde se produce el desarrollo de un producto que es entregable potencialmente.

También **se puede definir el Sprint como un mini proyecto**

### 3. Control: Burn Down

El Burn Down es la fase en la que **se mide el progreso de un determinado proyecto** Scrum. En ella, el Scrum Master será el encargado de actualizar los gráficos cuando se finalice cada uno de los Sprint.

Como conclusión, descubrimos que la metodología agile de Scrum es un proceso para llevar a cabo un conjunto de tareas de forma regular con el objetivo principal de trabajar de manera colaborativa. Y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras.




## Los principales beneficios que proporciona Scrum son:

-Entrega mensual (o quincenal) de resultados (los requisitos más prioritarios en ese momento, ya completados) lo cual proporciona las siguientes ventajas:
-Gestión regular de las expectativas del cliente y basada en resultados tangibles.
-Resultados anticipados (time to market).
-Flexibilidad y adaptación respecto a las necesidades del cliente, cambios en el mercado, etc.
-Gestión sistemática del Retorno de Inversión (ROI).
-Mitigación sistemática de los riesgos del proyecto.
-Productividad y Beneficios de Scrum calidad.
-Alineamiento entre el cliente y el equipo de desarrollo.
-Equipo motivado.

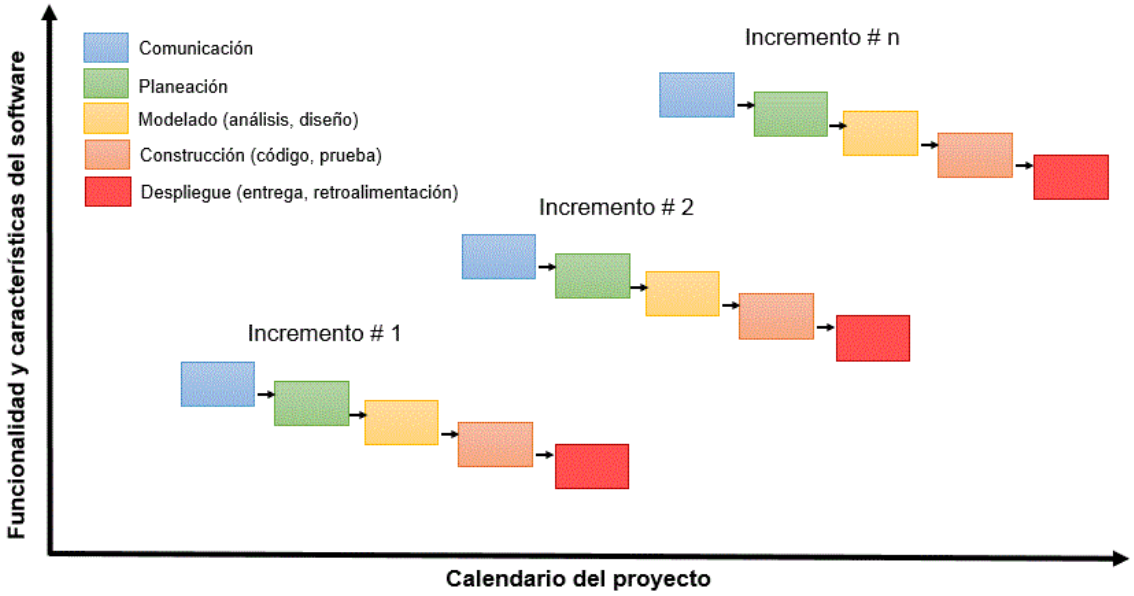
## ***Principales características de Scrum***

- **Gestión regular de las expectativas del cliente**, resultados anticipados, flexibilidad y adaptación, retorno de inversión, mitigación de riesgos, productividad y calidad, o, equipo motivado.
- **Se hace uso de equipos auto-dirigidos y auto-organizados.**
- **Se realiza a diario una reunión de Scrum**, que es una reunión de avance diaria que no dura más de 15 minutos con el objetivo de obtener realimentación sobre las tareas del equipo y los obstáculos que se presentan.

Cada uno de estos puntos mencionados hacen que el Scrum sea utilizado de manera regular en un conjunto de buenas prácticas para el trabajo en equipo y de esa manera obtener resultados posibles.

## Modelo de desarrollo incremental

En un desarrollo iterativo e incremental, el proyecto se planifica en diversos bloques temporales llamados iteraciones. De esta manera no se deja para el final del proyecto ninguna actividad arriesgada relacionada con la entrega de requisitos. En cada iteración el equipo evoluciona el producto a partir de los resultados completados en las iteraciones anteriores, añadiendo nuevos objetivos/requisitos o mejorando los que ya fueron completados. Un aspecto fundamental para guiar el desarrollo iterativo e incremental es la priorización de los objetivos/requisitos en función del valor que aportan al cliente.



**Ventajas:**

- 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

**Inconvenientes:**

Se requiere una experiencia importante para definir los incrementos y distribuir en ellos las tareas de forma proporcionada. Además podemos destacar los siguientes inconvenientes:

- 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.