


# Modelos y metodologías de desarrollo

## Modelos y metodologías de desarrollo

Las actividades de testing tienen sentido cuando se enmarcan en un ciclo de desarrollo del software. A este ciclo lo atraviesan distintas metodologías compuestas por etapas. En cada una se produce algún tipo de material.

Según cuál sea la metodología elegida, puede hablarse de ciclos de desarrollo **secuencial** o **iterativo**.

En este sentido las tareas de testing se organizan acorde a la metodología elegida.

 **¡Pro tip alert!** *Testeamos softwares en etapa de desarrollo que eventualmente tendrán su aprobación para iniciar la producción.*

## Ampliemos.. ¿Cómo comienza un ciclo de vida de desarrollo de software?

El ciclo de vida de desarrollo de software establece las etapas y procesos fundamentales que guían desde la concepción de una idea hasta la implementación y entrega del producto final.

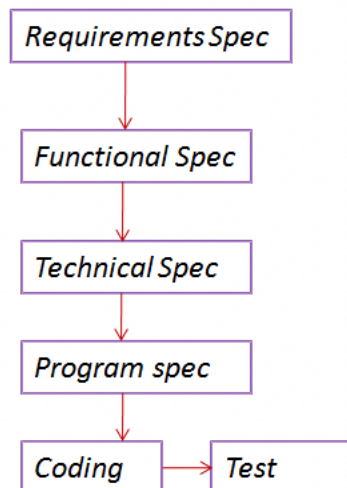
Las etapas principales del ciclo de vida son:

- ❖ Relevamiento de los requerimientos para el desarrollo del producto
- ❖ Expresión y documentación de cada requerimiento
- ❖ Desarrollo de código utilizando la documentación como insumo.
- ❖ Testing del producto

A continuación vamos a ver tres tipos de metodologías que se utilizan en la industria IT. La metodología adecuada a utilizar depende de la naturaleza y los requisitos del proyecto, así como de las preferencias y necesidades del equipo de desarrollo.

## **Metodología Cascada: Desarrollo en cascada – Waterfall model**

Esta metodología es de carácter secuencial: es decir que cada etapa inicia solo cuando la etapa anterior finalizó.



\*Fuente: <https://www.getsoftwareservice.com/v-model-of-testing/>

### **¿Y el testing?**

Se realiza solo cuando el código fue desarrollado por completo. El testing funciona como “evaluación de calidad” ya sea para aceptar o rechazar el producto.

### **Te dejamos aquí un ejemplo para su análisis.**

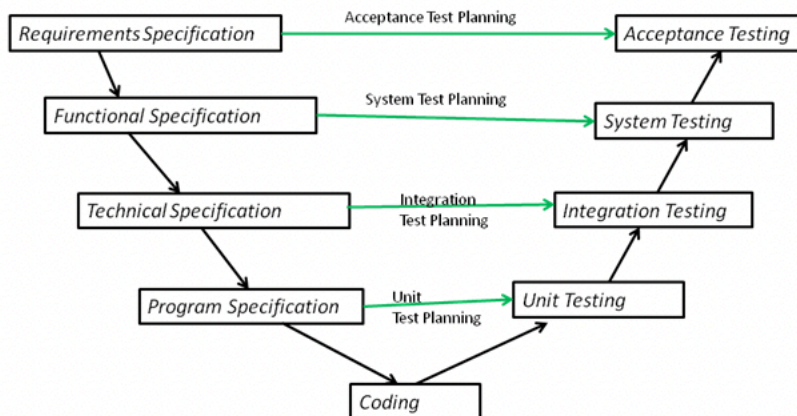
En una analogía con una fábrica, el testing podría representar el retiro de la línea de producción de aquellos productos con fallas para que solo salgan al mercado los productos que pasaron las pruebas de calidad.

En desarrollo de software no siempre es posible quitar parte del desarrollo. En primer lugar porque al hacerlo podríamos causar fallas en otras áreas. Además porque si lo que deberíamos retirar se encuentra cerca del final del ciclo y fuese parte de una funcionalidad clave, el sistema quedaría obsoleto o inaceptable su nivel de calidad.

## Modelo V

Este modelo al igual que el anterior, también es de carácter secuencial. Podría decirse que es una extensión del modelo en cascada ya que intenta resolver su desventaja principal: el testing está al final relanteciendo la posibilidad de encontrar defectos.

Como se puede observar en la siguiente imagen, a cada etapa del modelo en cascada le corresponden actividades de planificación relacionadas con la etapa en desarrollo.



Fuente: <https://www.getsoftwareservice.com/v-model-of-testing/>

Con este modelo se pueden identificar defectos al finalizar cada etapa.

Al igual que en el desarrollo en cascada, cada etapa debe cerrarse antes de comenzar la siguiente.

### **Te dejamos aquí un ejemplo para su análisis.**

La planificación del UAT (user acceptance testing) se genera cuando la especificación de los requerimientos están listos, claros y documentados.

Esto implica que omisiones, inexactitudes y todo tipo de errores humanos (que pueden dar lugar a defectos) son posibles de identificar y corregir al principio del ciclo, antes de continuar con las etapas siguientes.

## **Metodología Ágil: Modelos iterativos e incrementales**

Las metodologías ágiles producen software en entregas a lo largo de *iteraciones*, *ciclos* o *sprints* (tres palabras similares para referirnos a lo mismo).

En cada iteración se avanza en la construcción del sistema en producción.

Generalmente los ciclos o iteraciones son de 2, 3 o 4 semanas

Cada ciclo tiene un comienzo y un momento final delimitado. Usualmente las actividades que se realizan dentro del ciclo son las mismas.

Esta metodología de trabajo, permite que el software salga a producción con una versión del producto que puede modificarse a lo largo del tiempo o en los siguientes ciclos de desarrollo. Esto abre la posibilidad de recibir feedback por parte del usuario a lo largo de las iteraciones.

El producto se puede ajustar conforme modificaciones en el mercado o a las necesidades del cliente o usuario final. En ocasiones significa que el trabajo de la construcción del producto puede no tener fecha de finalización o que comunicar una fecha de cierre a un potencial cliente puede ser algo difícil.

Como desventaja puede mencionarse que en ciclos o iteraciones no se documenten correctamente los procesos. Para mitigar este posible error, es importante la tarea de análisis del equipo de testing y la implementación de TDD<sup>1</sup> como método de trabajo.

### **Te dejamos aquí un ejemplo para su análisis.**

Algunos Modelos iterativos son también conocidos como **Metodologías Ágiles**:

- **Scrum** – Una variante muy conocida, con iteraciones cortas y foco en la autoorganización y mentalidad de equipo. Toma su nombre del deporte Rugby por generar paralelismos entre la metodología de trabajo y el juego.
- **Kanban** – Hace foco en la visualización del flujo de trabajo para encontrar y eliminar los cuellos de botella.
- **RUP** – Rational Unified Process – Iteraciones un poco más largas que Scrum. Fases de transición: Incepción, elaboración, construcción.
- **Spiral** – El factor de riesgo se usa para determinar el nivel de documentación y esfuerzo dedicado a cada fase.

---

<sup>1</sup> TDD = Test Driven Development implica que los tests funcionales se escriben primero y el desarrollo de código viene después.

En síntesis, el testing, crucial para garantizar la calidad del producto, varía según la metodología adoptada, desde su ejecución al final del desarrollo en enfoques secuenciales hasta su integración continua en ciclos iterativos ágiles.