



CICLO 1

[FORMACIÓN POR CICLOS]

Fundamentos de **PROGRAMACIÓN**



**UNIVERSIDAD
DE ANTIOQUIA**

Facultad de Ingeniería

Lectura

ETAPAS DEL CICLO

de vida

```
def block(code):
    opened = []
    blocks = []
    for i in range(len(code)):
        if code[i] == '{':
            opened.append(i)
            code[i] == '}':
            blocks[i] = opened.pop()
        else:
            blocks[opened.pop()] = i
    return ''.join(c for c in code if c in '<=>{}')

def run():
    code = parse(code)
    x = 0
    b = 0
    while block(code) < 1:
        sym = code[i]
        if sym == '>':
            x += 1
            he.setdefault(x, 0)
        elif sym == '<':
            x -= 1
        elif sym == '+':
            bf[x] += 1
        elif sym == '-':
            bf[x] -= 1
        elif sym == ',':
            print(chr(bf[x]), end='')
        elif sym == ',':
            bf[x] = int(input('input: '))
        elif sym == '[':
            if not bf[x]: i = blocks[i]
        elif sym == ']':
            if bf[x]: i = blocks[i]
    code = input()
```

El ciclo de vida del desarrollo de *software* (CVDS) es un proceso por el cual un ingeniero produce *software* con la máxima calidad y el menor coste en el menor tiempo posible. El CVDS proporciona un flujo de fases bien estructurado que ayuda a una organización a producir rápidamente *software* de alta calidad, bien probado y listo para su uso en producción.

El CVDS consta de seis fases:

- Análisis de los requisitos
- Planificación
- Diseño de *software* o diseño arquitectónico
- Desarrollo de *software*
- Pruebas
- Despliegue

El CVDS funciona reduciendo el coste del desarrollo de *software* y, al mismo tiempo, mejorando la calidad y acortando el tiempo de producción. El CVDS consigue estos objetivos aparentemente incompatibles siguiendo un plan que elimina los obstáculos típicos de los proyectos de desarrollo de *software*. Ese plan comienza con la evaluación de los sistemas existentes en busca de deficiencias, luego define los requisitos del nuevo sistema y a continuación crea el software a través de las etapas de análisis, planificación, diseño, desarrollo, pruebas y despliegue. Al anticiparse a errores costosos, como no pedir la opinión del usuario final o del cliente, el CVDS puede eliminar la repetición de trabajos redundantes y las correcciones en etapas posteriores.

También es importante saber que hay un fuerte enfoque en la fase de pruebas. Como el CVDS es una metodología repetitiva, hay que garantizar la calidad del código en cada ciclo. Muchas organizaciones tienden a dedicar pocos esfuerzos a las pruebas, mientras que un mayor enfoque en las pruebas puede ahorrarles mucho trabajo, tiempo y dinero. Vale la pena ser previsor y escribir los tipos de pruebas adecuados.



Etapas del ciclo de vida

Hay distintas metodologías que siguen las etapas en distintos órdenes; sin embargo, todas llevan a cabo un proceso similar en cada una. Seguir las etapas del CVDS asegura que el proceso funcione de manera fluida, eficiente y productiva. Así pues, a continuación, daremos un pequeño esbozo de las actividades realizadas en cada una.

1. Análisis de requisitos

“¿Cuáles son los problemas actuales?”. Esta etapa del CVDS significa obtener información de todas las partes interesadas, incluidos los clientes, los vendedores, los expertos de la industria y los programadores. En esta fase se procede a analizar las necesidades que tienen los usuarios del futuro sistema *software* y que deben ser satisfechas mediante el funcionamiento de este. El cliente que solicita el desarrollo expone sus necesidades, requisitos que debe cumplir el *software*, y el ingeniero los recoge y analiza. De acuerdo con esto, el equipo elabora una especificación precisa del sistema a desarrollar.

Una vez tenemos claros los deseos del cliente para el producto, es necesario pasar por un proceso de planeación. Aquí, el equipo determina el coste y los recursos necesarios para implementar los requisitos analizados. También es necesario identificar los riesgos implicados y, a partir de estos, diseñar medidas para suavizar esos riesgos. En otras palabras, el equipo en este paso debe determinar si el proyecto es viable y cómo se puede implementar teniendo en cuenta el menor riesgo que se pudiese presentar.

2. Diseño

“¿Cómo vamos a conseguir lo que queremos?”. Esta fase consiste en elaborar un esquema o diseño en el que se contemplen los elementos necesarios para que el sistema funcione según con lo estipulado en el análisis, pero también debe establecerse un esquema y un orden del sistema para su construcción. Un adecuado diseño permite optimizar los recursos en la producción de este. El producto de esta fase consiste generalmente en una serie de diagramas en el lenguaje UML³ (este lenguaje se estudiará en más detalle en próximos ciclos) en los cuales se tienen en cuenta todos los pormenores técnicos necesarios para hacer realidad el deseo del cliente. Un fallo en esta fase supondrá, casi con toda seguridad, un sobrecoste en el mejor de los casos, ya que sería necesario rediseñar la aplicación, perdiéndose los avances hasta ese momento, y el colapso total del proyecto en el peor de los casos. Por tal motivo, el encargado de esta fase suele ser el arquitecto de *software* de la compañía, un ingeniero con vasta experiencia en el área, que en ocasiones también cumple la función del líder del proyecto.

³ Es el lenguaje de modelado de sistemas de *software* más conocido y utilizado en la actualidad; está respaldado por el Object Management Group. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

3. Codificación o desarrollo

"Vamos a crear lo que queremos". En esta fase comienza el desarrollo propiamente dicho. Se programará, por separado, cada uno de los elementos que fueron definidos en la fase de diseño utilizando para ello las herramientas que el equipo considere necesarias: lenguajes de programación, sistemas de bases de datos, sistemas de información, etc. Así mismo, se construirán los elementos necesarios para verificar que los desarrollos funcionan correctamente, los cuales se utilizan directamente en la próxima fase.

4. Pruebas del código e integración

"¿Hemos conseguido lo que queríamos?". En esta fase, después de desarrollados todos los componentes planteados, se procede a unirlos todos con el objetivo de construir un sistema completo y funcional. Luego, se prueban los defectos y las deficiencias, arreglándolos hasta que el producto cumpla con las especificaciones originales. En síntesis, con ello se verifica si el código cumple con los requisitos definidos y no cuenta con posibles fallos que comprometan el adecuado funcionamiento del producto.

5. Despliegue de software

"Vamos a empezar a usar lo que tenemos". Esta fase no forma parte per se del ciclo de desarrollo de *software*, aunque sí influye en el resto de las fases descritas. Comprende el período de funcionamiento de la aplicación y es el objetivo final del producto desarrollado. Su desempeño decidirá las acciones a tomar en fases posteriores de desarrollo como la de mantenimiento. En esta etapa se despliega el *software* en el ambiente de producción⁴ para que los usuarios puedan empezar a utilizar el producto. Sin embargo, muchas organizaciones optan por mover el producto a través de diferentes ambientes de despliegue, como el de calidad o producción. Esto permite a los interesados jugar con el producto de forma segura, lo que ayuda a detectar cualquier error antes de lanzarlo al mercado.

Fase extra: mantenimiento

Durante la fase de despliegue del *software*, a veces es necesario realizar cambios, bien para corregir errores no detectados en las fases de desarrollo o para introducir mejoras. Cualquier sistema que se ponga en funcionamiento durante un período, tendrá un feedback respecto a su diseño y funcionamiento, en el que en ocasiones se podrán observar problemas pasados por alto en la fase de diseño. Ante estas nuevas situaciones de funcionamiento el sistema debe evolucionar para responder a las nuevas demandas. Esta evolución se desarrolla en la fase de mantenimiento; sin embargo, dependiendo de la metodología de ciclo de vida usada por el equipo, este proceso puede pasar a realizarse por fuera. A continuación, hablaremos de estas metodologías.

⁴ Un ambiente (o entorno) es el sistema en que se instalará la aplicación, y sus características dependen de las necesidades de cada aplicación. Generalmente se trabaja con tres entornos: desarrollo, calidad y producción. En el ambiente de producción se despliega la aplicación para el consumo del público, y el entorno de calidad debe ser lo más similar posible a producción, ya que en este se realizan todas las pruebas necesarias para garantizar el debido comportamiento de la aplicación. El ambiente de desarrollo es el lugar donde el equipo despliega sus códigos mientras se construyen.