CICL

[FORMACIÓN POR CICLOS]

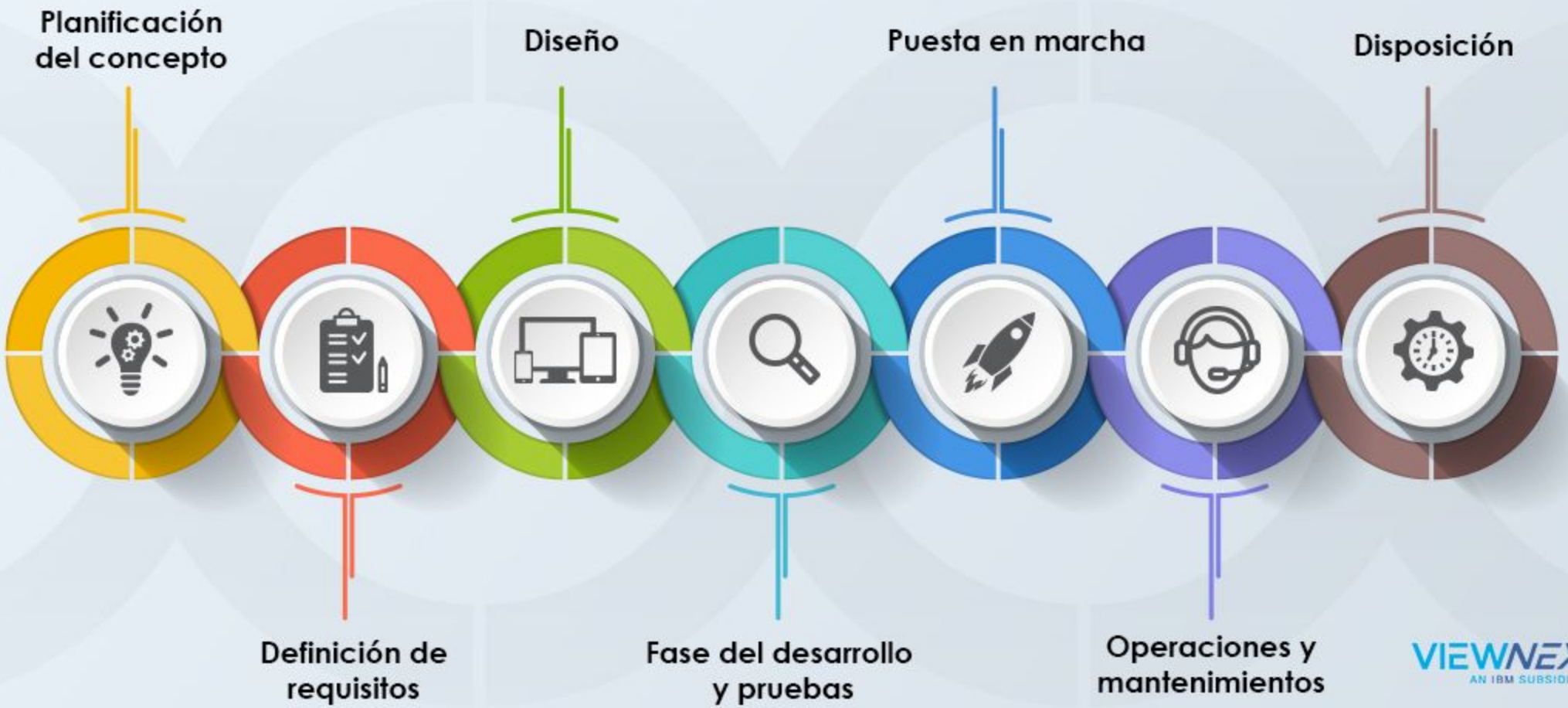
Programación Básic

Pruebas Unitarias

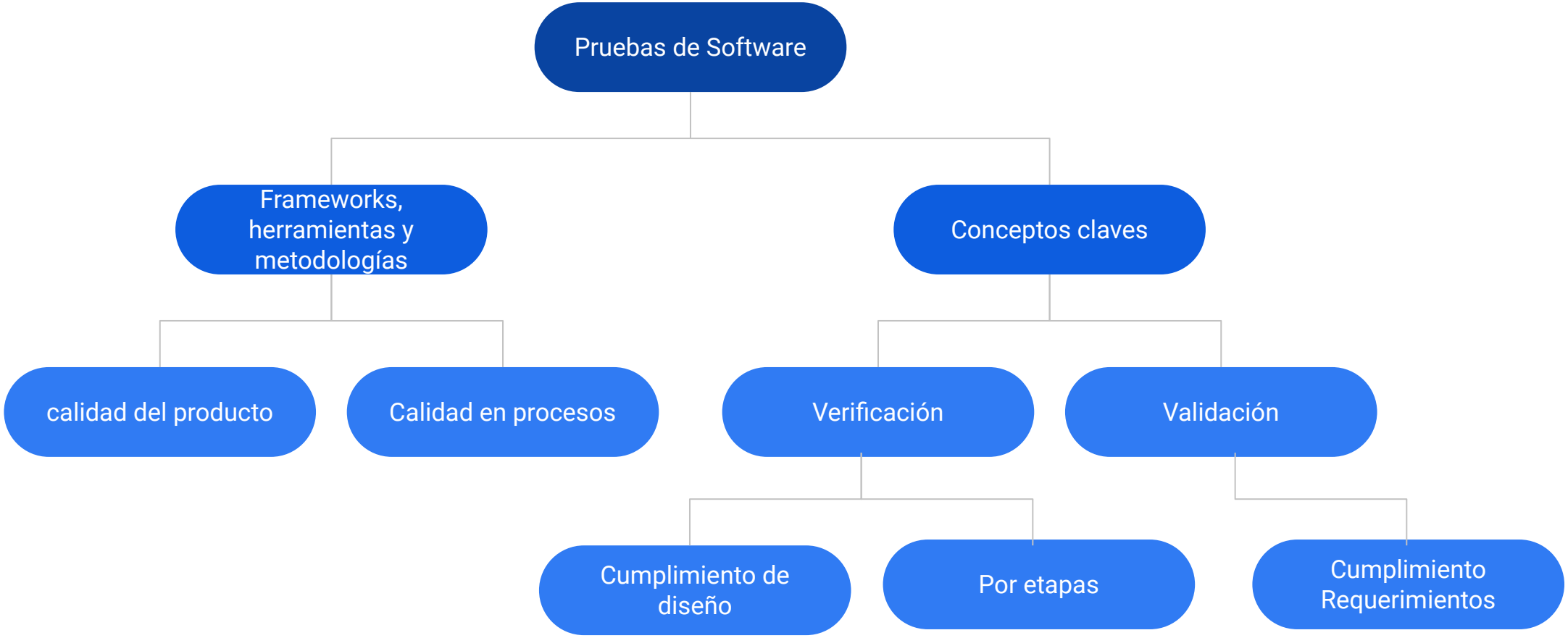
Ingeni@ Soluciones TIC

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Las 7 fases del SDCL



Introducción



Testing


01
Permite identificar errores durante las fases de desarrollo.
02
Asegura la confianza del usuario final y su satisfacción al usar el software.
03
Permite asegurar la calidad del producto y su correcto funcionamiento.
04
Es necesario para mantener la reputación del negocio en el sector.

Pruebas de aceptación

- Las pruebas de aceptación (o acceptance tests) están orientadas a la validación del sistema de software, y buscan **evaluar la conformidad** de este con los requisitos de negocio, valorando si es **aceptable para ser entregado** a clientes y usuarios.

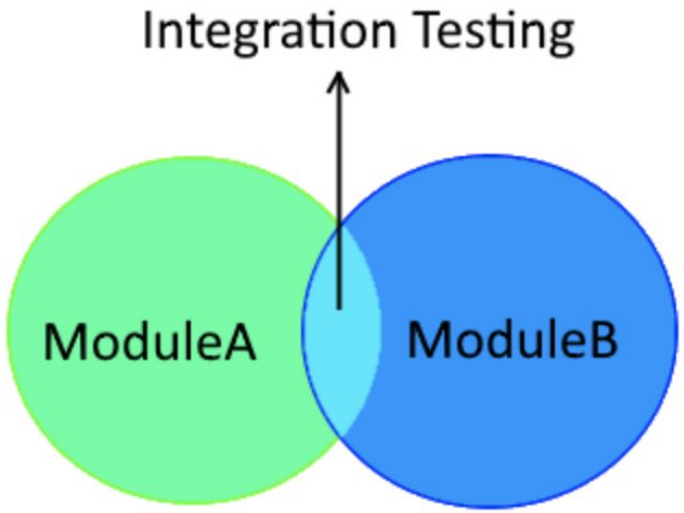
Pruebas de Sistema

- Buscan evaluar un sistema en su totalidad.
- El sistema suele estar terminado o próximo a su finalización
- Estas pruebas comprenden acciones que realizaría un usuario
  - Interacción del sistema con otros sistemas asociados.
- Son pruebas de caja negra
  - se realizan sin que se conozca el código fuente ni el funcionamiento interno del sistema en gran detalle.

Pruebas de Sistema

Pruebas de Integración

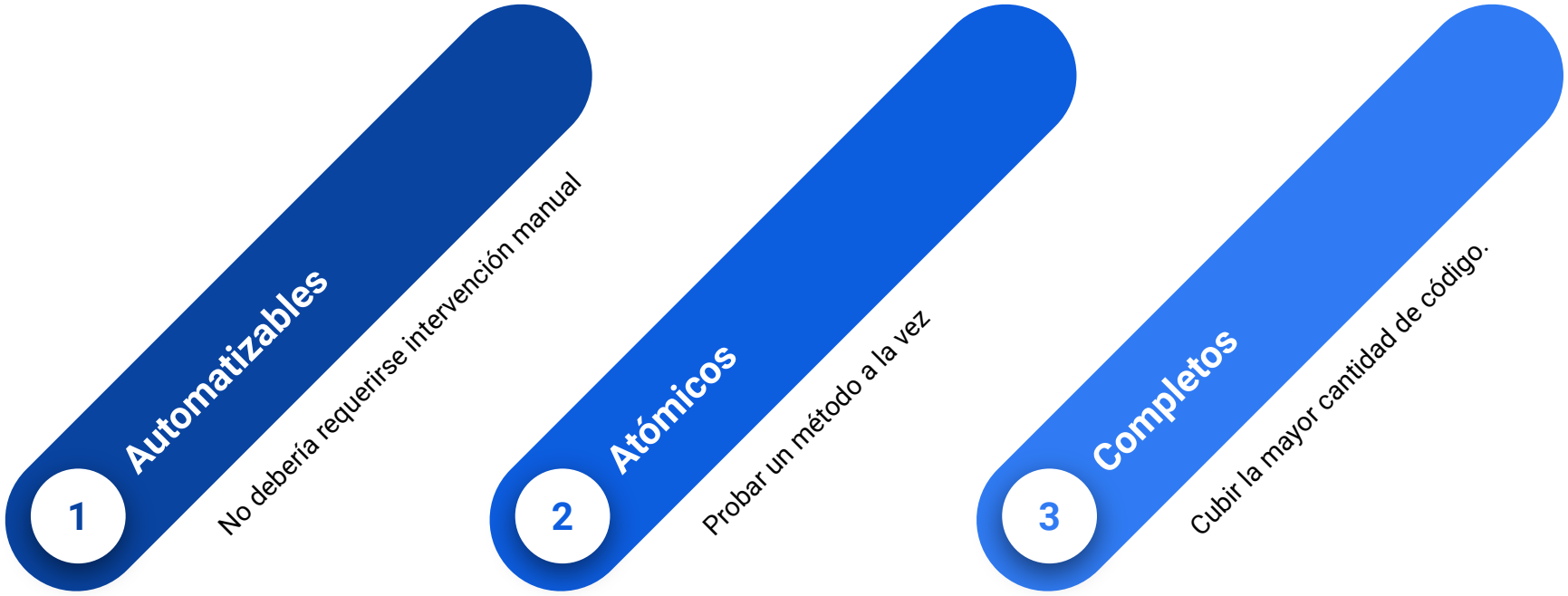
- Evaluar la integración, interacción o combinación entre dos o más componentes de código de un sistema de software.
- Es común que dos o más componentes de software trabajen juntos dentro de una aplicación y se generen dependencias entre ellos.
- Estas dependencias se deben verificar a medida que el sistema de software evoluciona.



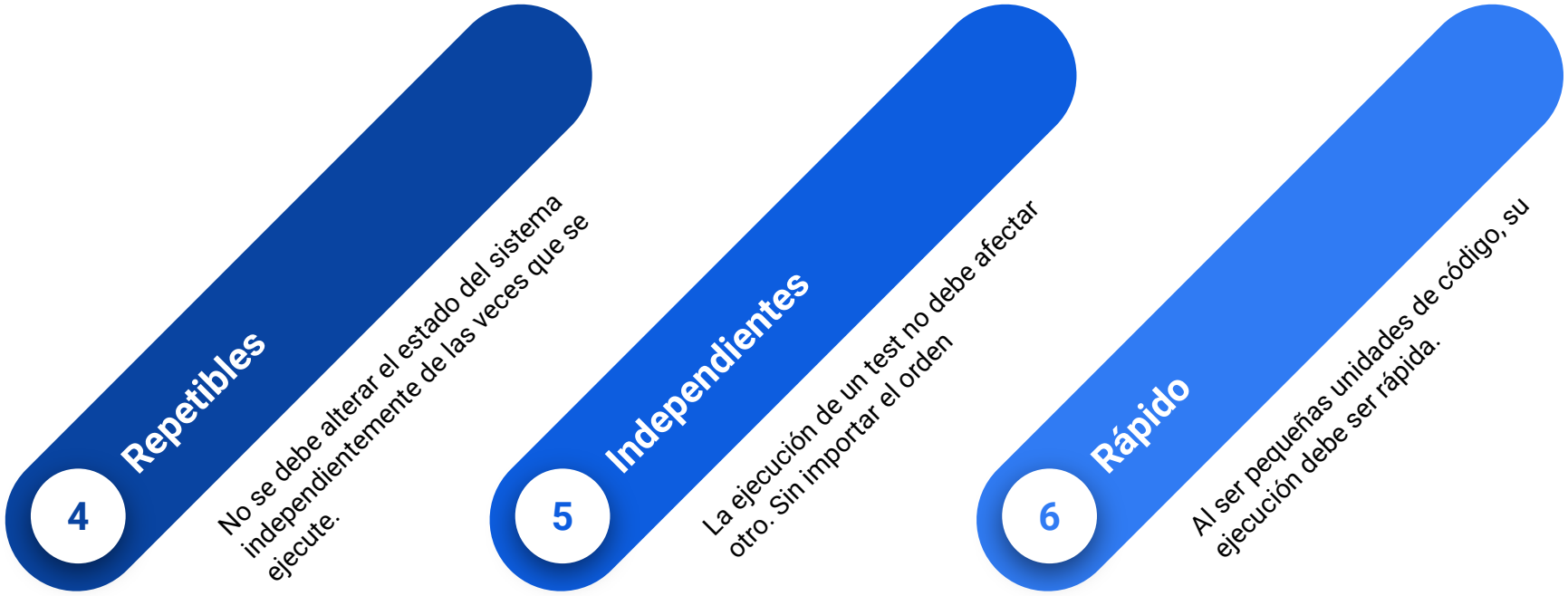
Pruebas Unitarias

- Evaluar el comportamiento de una clase, una función o un componente de manera independiente al resto del sistema de software o aplicación.
- Pruebas de caja blanca
  - pruebas en las cuales se tiene conocimiento de
    - código fuente de la aplicación
    - funcionamiento interno detallado.
- Las características de estas pruebas han permitido que sean ampliamente automatizadas.

Características



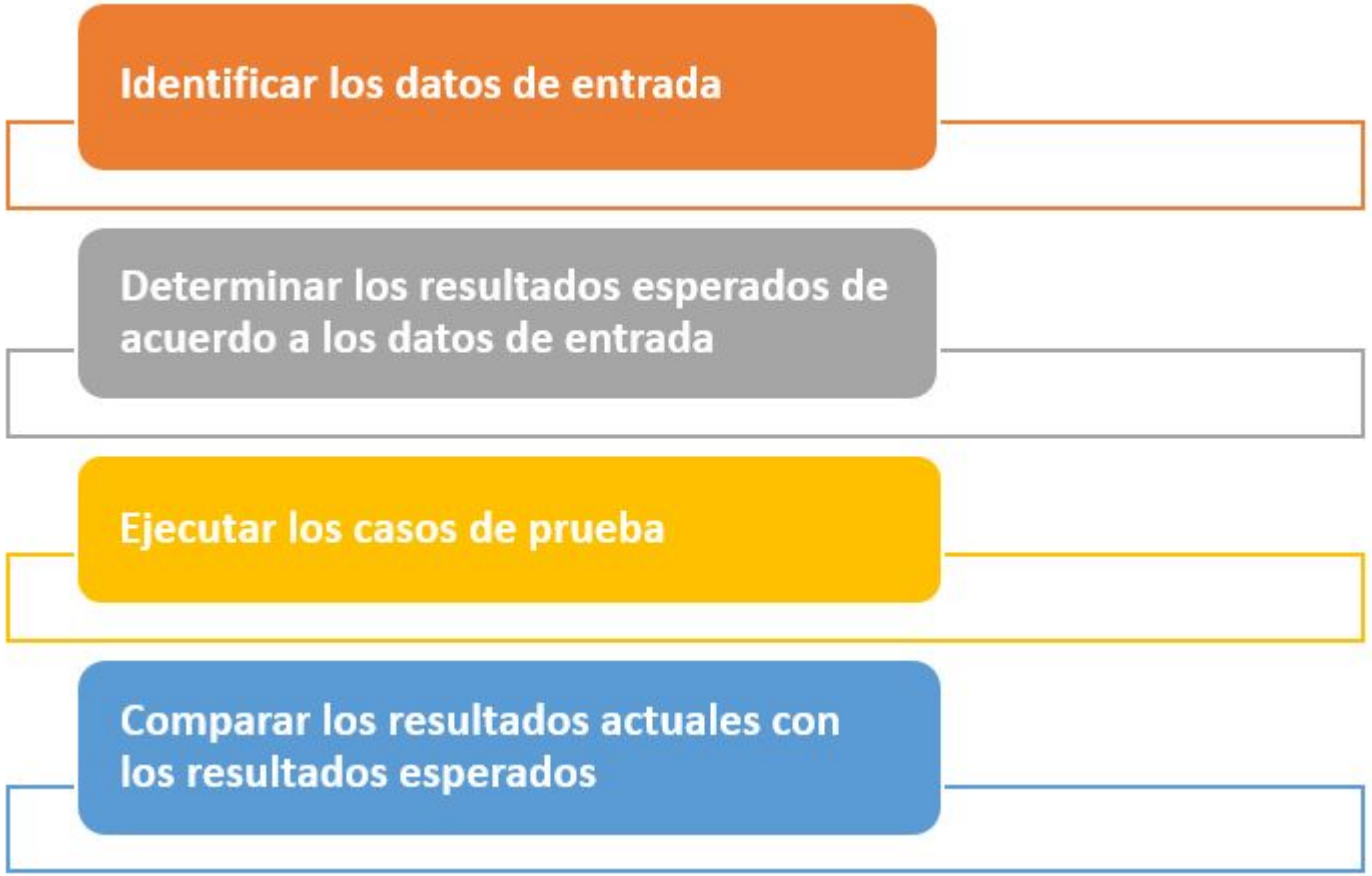
Características

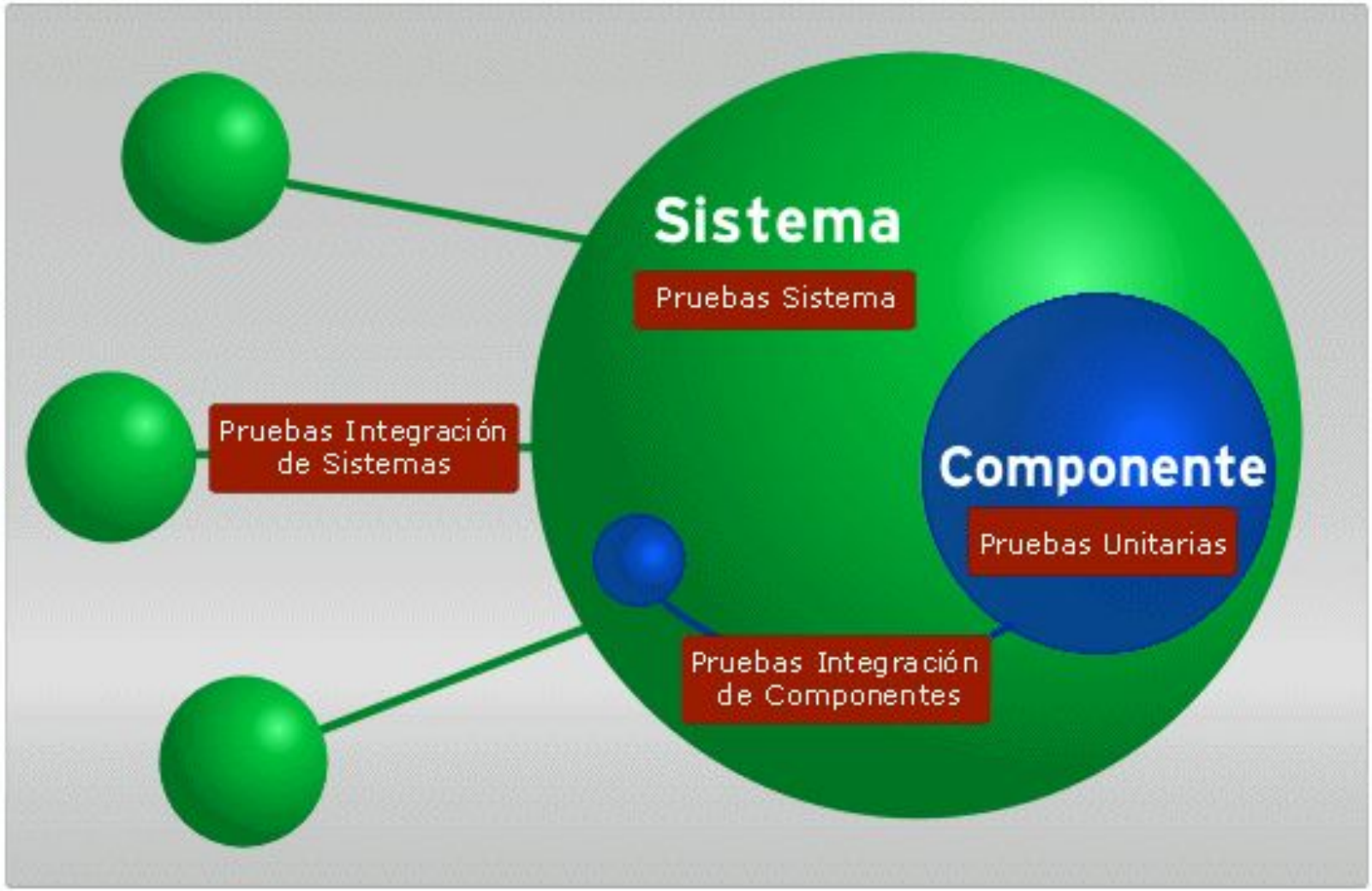


Un algoritmo es observable si:

- Tiene un tipo de retorno diferente de void.
- Es aislado:
  - No obtiene datos de fuentes externas como bases de datos o archivos
- No contienen ningún tipo de lógica
- Se pueden ejecutar en cualquier orden.

Pruebas Unitarias




Sistema
Pruebas Sistema
Componente
Pruebas Unitarias
Pruebas Integración de Sistemas
Pruebas Integración de Componentes


Pruebas unitarias
Despliegue ambiente QA
API
Pruebas de APIs e integración
Pruebas de performance y carga
Pruebas de UI
Despliegue a producción

Automatización de pruebas

- Herramientas de software para controlar la ejecución de prueba
  - Comparar los resultados obtenidos vs. resultados esperados
- Cuenta con algunos beneficios potenciales, tales como:
  - Reducir el costo de realización de pruebas.
  - Reducir los errores humanos al probar.
  - Reducir las diferencias en la calidad de las pruebas que se puede presentar entre diferentes individuos.
  - Reducir significativamente el costo de pruebas de regresión, es decir, de realizar la misma prueba de manera repetida a medida que el sistema de software evoluciona.

Plan de pruebas

- Pruebas de diferentes niveles que se realizarán en una aplicación o sistema de software.
- Incluye diferentes criterios que se deben tener en cuenta a la hora de realizar dichas pruebas.
- El plan de pruebas suele estar compuesto por uno o varios conjuntos de pruebas.
- Un conjunto de pruebas (o *set* de pruebas)
  - un conjunto de casos de prueba de un nivel de pruebas determinado (p.e., un conjunto de pruebas unitarias).
  - Tiene casos de prueba.

Casos de prueba

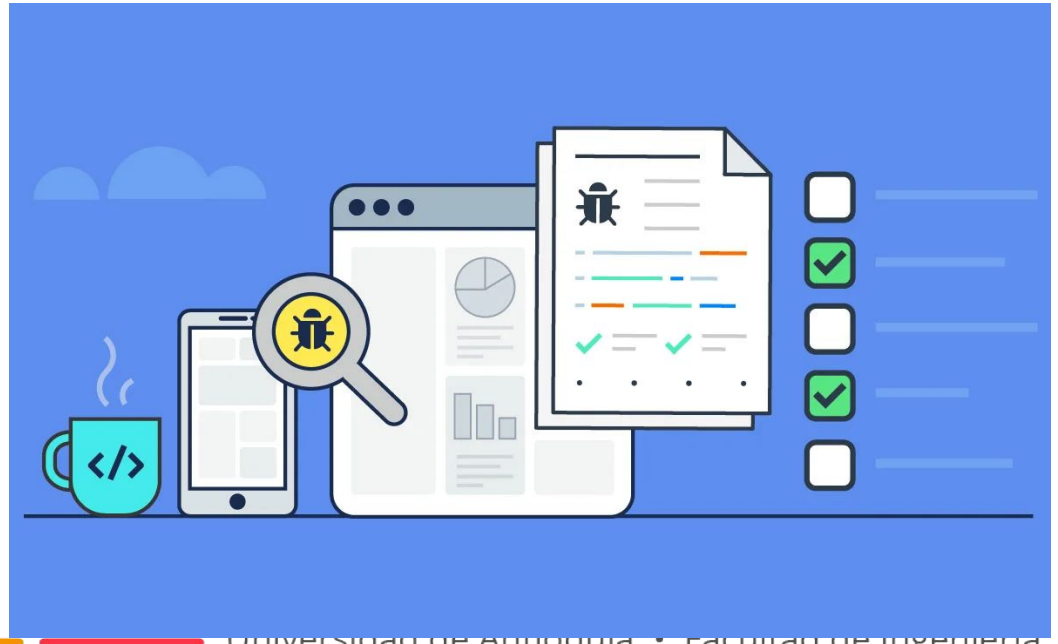
Un caso de prueba especifica y documenta una prueba que se le realiza a un componente o aplicación de software.

Por lo general, un caso de prueba tiene los siguientes componentes:

- Los valores de entrada o parámetros necesarios para la ejecución del caso de prueba en cuestión.
- Los resultados esperados son los valores, datos de salida y/o funcionamiento esperados de parte del componente o sistema si este se comporta como se espera de acuerdo a los valores de entrada.
- Los valores prefijados (o prefix values) son las entradas o parámetros necesarios para establecer el componente de software en un estado apropiado previo al inicio del caso de prueba, es decir, antes de recibir los valores de entrada mencionados arriba.
- Los valores postfijados (o postfix values) son las entradas a ingresar al sistema después de que reciba los valores de entrada, y que son necesarios para devolver el sistema a un estado deseado.

Script de pruebas

Es un caso de prueba preparado de tal forma que pueda ser ejecutado automáticamente usando un software de pruebas, produciendo un reporte de resultados. A algunos de estos softwares de pruebas se les denomina frameworks de automatización de pruebas.



Framework de automatización de pruebas

es un conjunto de conceptos y herramientas que soportan y posibilitan la automatización de pruebas, siendo **dependientes del lenguaje de programación.**

Por lo general, la mayor parte de dichos *frameworks* soportan:

- Aserciones para evaluar los resultados obtenidos vs. los resultados esperados.
- La posibilidad de compartir datos comunes entre diferentes pruebas.
- *Sets* o conjuntos de pruebas, para organizar y ejecutar pruebas más fácilmente.
- La posibilidad de ejecutar pruebas, tanto mediante línea de comandos, como mediante una interfaz gráfica de usuario.

Existe una gran variedad de *frameworks* de pruebas para el lenguaje Java, asociados a diferentes niveles

- Cucumber para pruebas de aceptación o JMeter para pruebas de carga
- En cuanto a las pruebas unitarias, el *framework* de automatización más popular es **JUnit.**

JUnit

es un *framework* de automatización de pruebas de código abierto para Java.

Al ser de código abierto

- es desarrollado, mantenido y mejorado por una comunidad de desarrolladores no comercial alrededor del mundo.
- Se usa para escribir y ejecutar pruebas unitarias y de integración, automatizadas y repetibles.
- Puede ser usado por separado
    - Es muy común que se utilice con la ayuda del **IDE** o de un **servidor de integración continua**.

JUnit

- Una prueba se implementa mediante un método de prueba.
- Las clases de prueba que incluyen uno o varios métodos de prueba.
- Una clase de prueba puede incluir métodos que preparen el estado del objeto a probar antes y/o después de realizarse las pruebas (implementando los *prefix values* y los *postfix values*).
- En JUnit, las pruebas se hacen, principalmente, mediante aserciones
  - Una aserción es una instrucción en la cual chequeamos que se cumpla una condición en la ejecución del código.
    - Si dicha condición se cumple (y la aserción es verdadera), la prueba “pasa”.
    - Si la condición no se cumple, la prueba “no pasa”.